Open access • Proceedings Article • DOI:10.1109/INFCOM.2000.832555

Adaptive proportional routing: a localized QoS routing approach — Source link

Srihari Nelakuditi, Zhi-Li Zhang, Rose P. Tsang

Published on: 26 Mar 2000 - International Conference on Computer Communications

Topics: Static routing, Routing domain, Policy-based routing, Routing protocol and Dynamic Source Routing

Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 02-029

Adaptive Proportional Routing: A Localized QoS Routing Approach

Srihari Nelakuditi, Zhi-li Zhang, Rose P. Tsang, and David Du

July 18, 2002

Adaptive Proportional Routing: A Localized QoS Routing Approach*

Srihari Nelakuditi†, Zhi-Li Zhang†, Rose P. Tsang‡ and David H.C. Du†

† Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN55455
{srihari,zhzhang,du}@cs.umn.edu

‡ Sandia National Laboratories
PO Box 969, Mail Stop 9011
Livermore, California 94550
rtsang@ca.sandia.gov

Abstract

Most of the QoS routing schemes proposed so far require periodic exchange of QoS state information among routers, imposing both communication overhead on the network and processing overhead on core routers. Furthermore, stale QoS state information causes the performance of these QoS routing schemes to degrade drastically. In order to circumvent these problems, we focus on *localized* QoS routing schemes where the edge routers make routing decisions using only "local" information and thus reducing the overhead at core routers. We first describe *virtual capacity based routing* (vcr), a theoretical scheme based on the notion of *virtual capacity* of a route. We then propose *proportional sticky routing* (psr), an easily realizable approximation of *vcr* and analyze its performance. We demonstrate through extensive simulations that adaptive proportional routing is indeed a viable alternative to the global QoS routing approach.

1 Introduction

Quality-of-Service (QoS) routing is concerned with the problem of how to select a path for a flow such that the flow's QoS requirements such as bandwidth or delay are *likely* to be met. In order to make judicious choices in path selection, it is imperative that we have some knowledge of the *global* network QoS state, e.g., the traffic load distribution in the network. In the design of any QoS routing scheme, we must therefore address the following two key questions: 1) how to obtain some knowledge of the global network state, and 2) given this knowledge, how to select a path for a flow. Solutions to these questions affect the performance and cost trade-offs in QoS routing.

*This paper was supported in part by NSF grant ANI-0073819 and NSF CAREER Award grant NCR-9734428. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. An earlier abridged version of this paper appeared in the Proceedings of IEEE INFOCOM'2000, March 2000.

1.1 QoS Routing: Global *vs.* Localized Approaches

The majority of QoS routing schemes [1, 7, 10, 20, 34, 38, 41] proposed so far require periodic exchange of *link QoS state* information among network routers to obtain a *global view of the network QoS state*. This approach to QoS routing is thus referred to as the *global* QoS routing approach. Because network resource availability changes with each flow arrival and departure, maintaining *accurate* network QoS state requires *frequent* information exchanges among the network nodes (routers). The prohibitive communication and processing overheads entailed by such frequent QoS state updates precludes the possibility of *always* providing each node with an *accurate* view of the current network QoS state. Consequently, *the network QoS state information acquired at a source node can quickly become out-of-date when the QoS state update interval is large relative to the flow dynamics*. Under these circumstances, exchanging QoS state information among network nodes is superfluous. Furthermore, path selection based on a *deterministic* algorithm such as Dijkstra's shortest path algorithm, where *stale QoS state information is treated as accurate*, does not seem to be judicious. In addition, the global view of the network QoS state may lead to the so-called *synchronization problem*: after one QoS state update, many source nodes choose paths with shared links because of their perceived available bandwidth, therefore causing over-utilization of these links. After the next QoS state update, the source nodes would avoid the paths with these shared links, resulting in their under-utilization. This oscillating behavior can have severe impact on the system performance, when the QoS state update interval is large. Due to these drawbacks, it has been shown that when the QoS update interval is large relative to the flow dynamics, the performance of global QoS routing schemes degrades significantly [1, 25, 34]. Though there have been some remedial solutions proposed in [10, 1, 2] to deal with the inaccuracy at a source node, the fundamental problem is still not completely eliminated.

As a viable alternative to the global QoS routing schemes, in [25, 26] we have proposed *a localized* approach to QoS routing. Under this approach, *no global QoS state information exchange among network nodes is needed*. Instead, source nodes infer the network QoS state based on flow blocking statistics collected *locally*, and perform flow routing using this *localized* view of the network QoS state. The proposed localized QoS routing approach has several advantages. First of all, without the need for global information exchange, the communication overhead involved is minimal. Second, core routers (i.e., non-source routers) do not need to keep and update any QoS state database necessary for global QoS routing, thereby reducing the processing and memory overhead at core routers. Last but not the least, the localized QoS routing approach does not require any modification or extension to existing routing protocols such as OSPF. Only source routers need to add a QoS routing enhancement to the existing routing module. This makes localized QoS routing schemes readily deployable with relatively low cost.

1.2 Adaptive Proportional Routing: A Localized Approach

The fundamental question in the design of a localized QoS routing scheme is *how to perform path selection based solely on a local view of the network QoS state so as to minimize the chance of a flow being blocked as well as to maximize the overall system resource utilization.* The problem of path selection in localized QoS routing is com-

plicated by many factors. For example, due to complex network topology, paths between many source-destination pairs may have shared links whose capacity and load are unknown to the sources. Furthermore, the network load can fluctuate dynamically, which can make a previously unloaded link suddenly overloaded. In addition, path selection decision made by one source may affect the decision of another source.

To effectively address these difficulties, we study a novel *adaptive proportional routing* approach for designing localized QoS routing schemes. Here we assume that the *path-level* statistics, such as the number of flows blocked, is the only available QoS state information at a source. Based on these statistics, adaptive proportional routing attempts to proportionally distribute the load from a source to a destination among multiple paths according to their perceived quality (e.g., observed flow blocking probability). In other words, adaptive proportional routing exploits the inherent randomness in path selection by proportioning flows among multiple paths. This is fundamentally different from the conventional, *deterministic* path selection algorithms (e.g., Dijkstra shortest path algorithm) used in global routing schemes, which always choose the "best" feasible path to route a flow. As a result, adaptive proportional routing effectively avoids the synchronization problem associated with global QoS routing schemes.

There are three major objectives in our investigation of adaptive proportional routing: *adaptivity*, *stability* and *simplicity*. With only a localized view of the network QoS state, it is important to adjust flow proportions along various paths adaptively in response to the dynamically changing network load. Stability is essential to ensure efficient system resource utilization and thus the overall flow throughput. Lastly, we are interested in employing *simple local rules and strategies at individual sources* to achieve adaptivity and ensure stability.

Towards these goals, we present a theoretical framework for studying adaptive proportional routing. Using Erlang's Loss Formula, we introduce the notion of *virtual capacity* which provides a mathematical framework to model multiple paths between a source and a destination, as well as to compute flow proportions based on locally observed flow blocking probabilities. We also introduce a *self-refrained* alternative routing method to deal with the potential "knock-on" effect in QoS routing. By incorporating this *self-refrained* alternative routing method into the virtual capacity model, we design a theoretical adaptive proportional routing scheme which allows source nodes in a network to adaptively adjust their flow proportions based solely on locally observed flow blocking statistics. Through numerical examples we demonstrate the desired *self-adaptivity* of this theoretical adaptive proportional routing scheme in achieving an eventual equilibrium system state. As a simple and practical implementation of the theoretical scheme, we present a scheme, *proportional sticky routing* (psr), which preserves the self-adaptivity of the theoretical scheme while avoiding its computational overhead. Finally, comparison of the *psr* scheme with the well-studied global QoS routing scheme, the *widest shortest path* (wsp) scheme, is made using simulations. These simulation results demonstrate that with its low overhead and comparable performance, a simple and easy-to-implement localized QoS routing scheme such as *psr* provides a viable alternative to a global QoS routing scheme such as *wsp*.

The remainder of the paper is organized as follows. Section 2 presents a theoretical framework for studying adaptive proportional routing. Section 3 describes the *psr* scheme, and simulation results are shown in Section 4. In Section 5, the related work is presented. Section 6 concludes the paper.

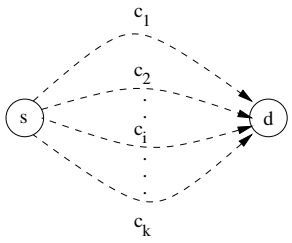


Figure 1: A set of disjoint paths between a source and a destination

2 Adaptive Proportional Routing: A Theoretical Framework

In all the QoS routing models we consider in this paper we assume that *source routing* (also referred to as *explicit routing*) is used. More specifically, we assume that the network topology information is available to all source nodes (e.g., via the OSPF protocol), and one or multiple *explicit-routed* (label switched) paths are set up *a priori* for each source and destination pair using, e.g., MPLS [32]. Flows arriving at a source to a destination are routed along one of the explicit-routed paths (hereafter referred to as the *candidate* paths between the source-destination pair). For simplicity, we assume that all flows have the same bandwidth requirement — one unit of bandwidth[1]. When a flow is routed to a path where one or more of the constituent links have no bandwidth left, this flow will be blocked. The performance metric in our study will be the overall blocking probability experienced by flows. We assume that flows from a source to a destination arrive randomly with a Poisson distribution, and their holding time is exponentially distributed. Hence the *offered* traffic load between a source-destination pair can be measured as the product of the average flow arrival rate and holding time. Given the offered traffic load from a source to a destination, the task of proportional QoS routing is to determine how to distribute the load (i.e., route the flows) among the paths between the source and destination (if there is more than one such path) so as to minimize the overall blocking probability experienced by the flows.

In this section, we first describe how to proportion the load among multiple paths when all the paths between the source and the destination are mutually disjoint. The notion of virtual capacity of a path is introduced to deal with sharing of links between different paths. A localized trunk reservation method is proposed to address the potential "knock-on" effect in QoS routing. We then present a theoretical adaptive proportional routing scheme that incorporates this *self-refrained* alternative routing method into the virtual capacity model.

2.1 An Idealized Proportional Routing Model

Consider a simple *fork* topology shown in Figure 1, where a source s and a destination d are connected by k *disjoint* paths $r_1, r_2, \ldots, r_k$. Each path r_i has a (bottleneck) capacity of c_i units of bandwidth, and is assumed to be known

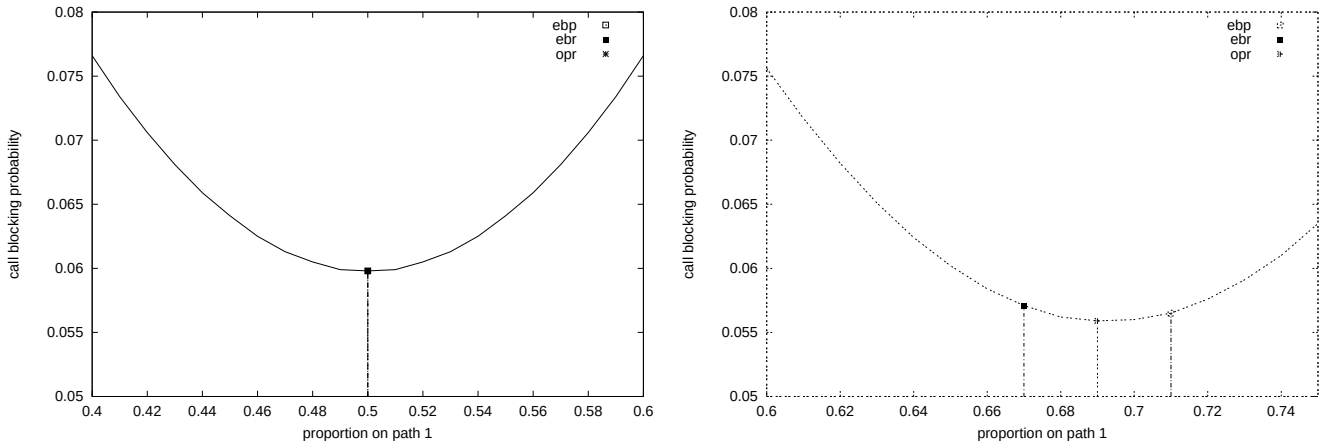
[1]The models presented in this paper can be extended to the case where flows have different bandwidth requirements using the extended Erlang loss formula [17, 30]. In Section 4, we conduct a simulation study of our localized QoS routing scheme using flows with heterogeneous bandwidth requirements.

to the source s. Suppose flows arrive at the source s at an average rate λ, and the average flow holding time is $1/\mu$. Throughout this section, we assume that flow arrivals are Poisson, and flow holding times are exponentially distributed. For simplicity, we also assume that each flow consumes 1 unit of bandwidth. In other words, path r_i can accommodate c_i flows at any time. *Without precise knowledge of the QoS state of a path (i.e., the available bandwidth of the path)*, a flow routed along the path has a certain probability of being blocked. Therefore, the question is how to route flows along these k paths so that the overall blocking probability is minimized. This problem can be formulated using the classic Erlang's Loss Formula as follows.

Suppose that, on the average, the proportion of flows routed along path r_i is α_i, where $i = 1, 2, \ldots, k$, and $\sum_{i=1}^{k} \alpha_i = 1$. Then the blocking probability b_i at path r_i is given by $b_i = E(\nu_i, c_i) = \frac{\frac{\nu_i^{c_i}}{c_i!}}{\sum_{n=0}^{c_i} \frac{\nu_i^n}{n!}}$, where $\nu_i = \alpha_i \frac{\lambda}{\mu}$ is referred to as the (average) load on path i. The total load on the system is denoted by $\nu = \sum_{i=1}^{k} \nu_i = \frac{\lambda}{\mu}$. To minimize the overall blocking probability, the *optimal* routing strategy (in the absence of precise knowledge of QoS state of each path) is therefore to route α_i^* proportion of flows along path r_i, $i = 1, 2, \ldots, k$, such that $\sum \alpha_i^* = 1$ and $\sum \nu \alpha_i^* b_i^*$ is minimized. This *optimal proportional routing* (*opr*) strategy can be implemented, for example, by routing flows to path r_i with probability α_i^*.

Given the total load ν and the path capacities c_i's, the optimal proportions α_i^*'s can be computed using an iterative search technique (e.g., hill-climbing) starting with a set of arbitrary proportions. For $k > 2$, the procedure of computing the optimal proportions is generally quite complex to implement in practice. To circumvent this problem, we consider two alternative strategies for flow proportioning: *equalization of blocking probabilities* (ebp) and *equalization of blocking rates* (ebr). The objective of the *ebp* strategy is to find a set of proportions $\{\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_k\}$ such that flow blocking probabilities of all the paths are equalized, i.e., $\tilde{b}_1 = \tilde{b}_2 = \cdots = \tilde{b}_k$, where $\tilde{b}_i$ is the flow blocking probability of path r_i, and is given by $E(\tilde{\alpha}_i \nu, c_i)$. The intuition behind *ebp* strategy is that if blocking probability b_i of a path r_i is greater than blocking probability b_j of a path r_j ($b_i > b_j$), then we can minimize the overall blocking probability by shifting some load from r_i to r_j. This increases b_j and decreases b_i and equilibrium state is reached when they are equal. On the other hand, the objective of the *ebr* strategy is to find a set of proportions $\{\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_k\}$ such that *flow blocking rates* of all the paths are equalized, i.e., $\hat{\alpha}_1 \hat{b}_1 = \hat{\alpha}_2 \hat{b}_2 = \cdots = \hat{\alpha}_k \hat{b}_k$, where $\hat{b_i}$ is the flow blocking probability of path r_i, and is given by $E(\hat{\alpha}_i \nu, c_i)$. The rationale behind *ebr* strategy is to assign a proportion α_i to a path r_i such that α_i is inversely proportional to blocking probability b_i along path r_i, i.e., $\alpha_i \propto \frac{1}{b_i}$. This results in equalization of blocking rates.

Unlike the optimal proportions, α_i^*'s, the proportions of *ebp*, $\tilde{\alpha}_i$'s, and those of *ebr*, $\hat{\alpha}_i$'s, can be computed using a simple iterative procedure starting with any arbitrary proportions. For example, consider the *ebp* strategy. Suppose we start with an initial set of proportions $\alpha_1^{(0)}, \alpha_2^{(0)}, \cdots, \alpha_k^{(0)}$. Let the corresponding blocking probabilities be $b_1^{(0)}, b_2^{(0)}, \cdots, b_k^{(0)}$, where $b_i^{(0)} = E(\alpha_i^{(0)} \nu, c_i)$. If $b_i^{(0)}$'s are all equal, then $\alpha_i^{(0)}$'s are the desired proportions. Otherwise, we use the mean blocking probability over all the paths, $\bar{b}^{(0)} = \sum b_i^{(0)}/k$, as the target blocking probability for each path, and obtain a new set of proportions, $\alpha_i^{(1)}$'s. The new proportions $\alpha_i^{(1)}$'s are computed from the Erlang's Loss Formula as follows: for $i = 1, 2, \ldots, k$, find the new load on path r_i, $\nu_i^{(1)}$, such that $\bar{b}^{(0)} = E(\nu^{(1)_i}, c_i)$. Then



(a) $c_1 = 10, c_2 = 10, \nu = 13$

(b) $c_1 = 20, c_2 = 10, \nu = 22$

Figure 2: Convergence points of *opr*, *ebp*, *ebr*

$\alpha_i^{(1)} = \frac{\nu_i^{(1)}}{\sum_{j=1}^{k} \nu_j^{(1)}}$. This procedure is repeated iteratively until we obtain a set of proportions such that the corresponding blocking probabilities are equal. Since for a fixed c_i the blocking probability b_i is an increasing function of its load $\alpha_i \nu$, it can be shown that the above iterative procedure will always converge. In the case of the *ebr* strategy, a similar iterative procedure can be used to obtain a set of proportions which equalize the blocking rates of all the paths.

Figure 2 shows the convergence points of the *ebp* and *ebr* strategies along with the *opr* strategy for a source and destination pair with two disjoint paths. Figure 2(a) shows the case where the capacities are equal. As expected, in this case all three strategies give equal proportions for the two paths. However, when the capacities are not equal as is the case shown in Figure 2(b), the equilibrium proportions for the two paths under the three strategies are different. It can be observed, however, the overall blocking probabilities under the *ebp* and *ebr* strategies are both quite close to that of the optimal strategy. Since it is generally computationally cumbersome to find the optimal equilibrium proportions, in this paper we will explore the two simple strategies, *ebp* and *ebr*, in adaptive proportional routing.

Before we leave this subsection, we would like to point out an interesting fact. In the network model shown in Figure 1, if we assume that source s has the precise knowledge of the QoS state (i.e., the available bandwidth) of each path at any given time, it can be shown that the overall flow blocking probability is given by $b = E(C, \nu)$, where $C = \sum_{i=1}^{k} c_i$. In other words, in terms of the overall flow blocking probability, the precise knowledge about the availability of paths makes it equivalent to the case where there exists a single path from source s to destination d with a capacity of $\sum_{i=1}^{k} c_i$. Due to multiplexing gain, blocking probability using multiple paths, even with optimal proportions, would be larger than using single path with the same aggregate capacity, i.e., $\sum_{i=1}^{k} \alpha_i^* E(c_i, \alpha_i^* \nu) > E(C, \nu)$ for $k > 1$, where α_i^*'s are optimal proportions. This fact illustrates the inherent performance loss due to not having the precise path QoS state information.

2.2 Virtual Capacity Model

In the idealized proportional routing model described above, we have assumed that all paths between a source and a destination are disjoint and their bottleneck link capacities are known. In practice, however, paths between a source and a destination have shared links. These paths may also share links with paths between other source-destination pairs. Furthermore, as traffic patterns across a network change, the bottleneck link of a path and its (perceived) capacity may also change. In order to address these issues, we introduce the notion of *virtual capacity* (vc) of a path.

Consider a source-destination pair. We model each path between them as one direct *virtual link* with a certain amount of capacity, referred to as the *virtual capacity* of the path. This virtual capacity is a function of *the load offered by the source along the path and the corresponding blocking probability observed by the source*. Formally, consider a path r between a source and a destination. Suppose a load of ν_r is offered by the source along the path, and the corresponding blocking probability observed by the source is b_r. Then the virtual capacity of the path, denoted by vc_r, is given by $vc_r = E_{vc}^{-1}(\nu_r, b_r)$, where $E_{vc}^{-1}(\nu_r, b_r)$ denotes the inverse function of the Erlang's Loss Formula[2] with respect to the capacity, and is given by

$$vc_r = E_{vc}^{-1}(\nu_r, b_r) := \min\{c \geq 0 : E(\nu_r, c) \leq b_r\}$$

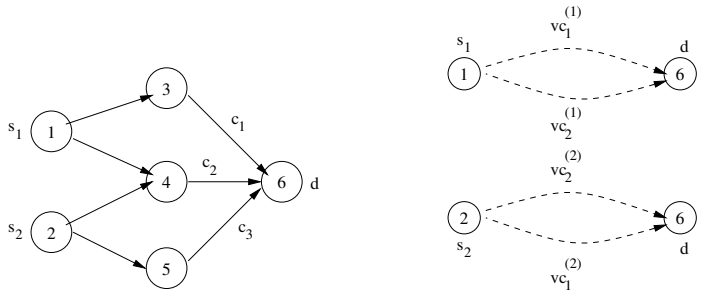
The notion of virtual capacity provides a mathematical framework to deal with shared links among multiple paths. For example, suppose m paths, $r_1, r_2, \ldots, r_m$, share a *bottleneck* link with capacity c. Then the virtual capacity vc_i of path r_i represents its "capacity share" on the bottleneck link. Let ν_i denote the offered load on the bottleneck link from path r_i. Then the blocking probability on the bottleneck link is given by $b = E(\sum_{i=1}^{m} \nu_i, c)$. Since flows routed along any of these m paths have the same probability to be blocked at the bottleneck link, the virtual capacity of path r_i is given by $vc_i = E_{vc}^{-1}(\nu_i, b_i) = E_{vc}^{-1}(\nu_i, b)$, where b_i denotes the observed blocking probability of path r_i. In particular, for path r_i, the larger the offered load ν_i is, the larger is its virtual capacity vc_i. This reflects the larger "capacity share" of the bottleneck obtained by path r_i because of its higher offered load[3].

Based on this notion of virtual capacity, we can model paths between a source and a destination as if they were all disjoint and had bottleneck capacities equal to their virtual capacities, as in the idealized proportional routing model (Figure 1). Unlike the idealized proportional routing model, however, the virtual capacity of a path is not fixed, but is a function of its offered load and the corresponding blocking probability. Since the virtual capacity of a path depends only on local statistics at a source (i.e., the offered load by a source and the corresponding blocking probability observed by the source), *flow proportioning based on virtual capacities of paths does not require any global QoS state information exchange*.

A key feature of our virtual capacity model is its *self-adaptivity*: proportions of flows (and therefore offered loads) along different paths between a source and a destination will be adjusted based on the observed blocking

[2]Note that $E_{vc}^{-1}(\nu_r, b_r)$ defined above is an integer-valued function. A *continuous* version of the Erlang's Loss Formula and its inverse functions can be defined [8] and used instead. For more details, the interested reader is referred to [27].

[3]It is also worth noting that $\sum_{i=1}^{m} vc_i \geq c$. This is due to "loss in multiplexing gain" when a shared channel is divided into multiple "dedicated" channels. To ensure the same blocking probability, the total capacity of the dedicated channels has to be larger than the capacity of the shared channel.



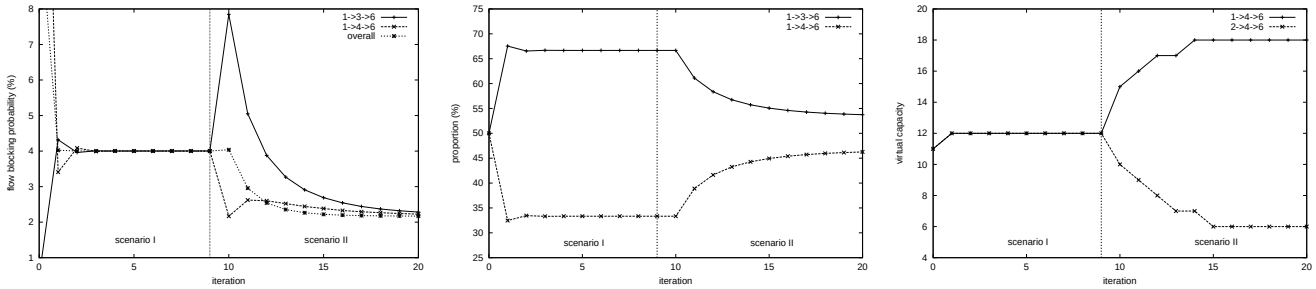
(a) physical view (b) virtual view

Figure 3: Illustration of virtual capacity model using *kite* topology

probability of those paths, an important measure of the "quality" of a path. From the definition of virtual capacity, we observe that for two paths with the same offered load, the path with higher observed blocking probability has lower virtual capacity. Therefore, if we are to equalize the observed blocking probabilities or blocking rates along these two paths, more flows should be routed to the path with lower observed blocking probability (and higher virtual capacity). The new proportions for these two paths can be computed based on their virtual capacities, as in the idealized proportional routing model.

We illustrate the self-adaptivity of the virtual capacity model through an example. Consider the *kite* topology shown in Figure 3(a), where two sources, s_1 and s_2, have two paths each to destination d, and two of the paths share a common link ($4 \rightarrow 6$). The links with labels are the bottleneck links of the network, where $c_1 = c_2 = c_3 = 20$, and all the other links can be viewed to have infinite capacities (i.e., flows are never blocked on these links). Let r_1^1, r_2^1 denote the paths $1 \rightarrow 3 \rightarrow 6$ and $1 \rightarrow 4 \rightarrow 6$ respectively, and r_1^2, r_2^2 denote the paths $2 \rightarrow 5 \rightarrow 6$ and $2 \rightarrow 4 \rightarrow 6$ respectively. The *virtual capacity view* of the two source-destination pairs are shown in Figure 3(b), where the paths r_2^1 and r_2^2 appear to each source as if they were disjoint with capacities vc_2^1 and vc_2^2 respectively. Note that if a path doesn't share links with any other path, its virtual capacity is the same as its actual bottleneck link capacity.

First consider the scenario where both sources have an offered load of 22. Suppose initially each source proportions flows *equally* between its two paths, i.e., $\nu_j^i = 11$, $i, j = 1, 2$. The blocking probabilities observed on paths r_1^1, r_2^1, r_2^2 and r_1^2 are $b_1^1 = 0.0046$, $b_2^1 = 0.2090$, $b_2^2 = 0.2090$, and $b_1^2 = 0.0046$ respectively, resulting in an overall blocking probability of 0.1068. The corresponding virtual capacities are $vc_1^1 = 20$, $vc_2^1 = 12$, $vc_2^2 = 12$, and $vc_1^2 = 20$. In particular, we see that the shared link of paths r_2^1 and r_2^2 is treated by each source as an exclusive link with capacity 12. For both sources, since the blocking probability of path r_2^i is much higher than path r_1^i, more flows will be proportioned to path r_1^i, as it has a larger virtual capacity vc_1^i. The new proportions can be computed based on the virtual capacities of the paths, using either the *ebp* strategy or the *ebr* strategy. For example, using the *ebp* strategy, the adaptation process for source s_1 is shown on the *left* side (scenario I) of Figure 4(a), where we see that after a few iterations the flow blocking probabilities of both paths r_1^1 ($1 \rightarrow 3 \rightarrow 6$) and r_2^2 ($1 \rightarrow 4 \rightarrow 6$) are equalized at around 0.04. Figure 4(b) shows the corresponding proportions of flows routed along these two paths during this



(a) *changes in blocking probabilities* (b) *changes in proportions* (c) *changes in vc's*

Figure 4: Adaptation process of *ebp*

adaptation process, where we see that source s_1 backs off from the path (r_2^1) with the shared bottleneck link $4 \rightarrow 6$, and directs more flows to the other path (r_1^1). The resulting flow proportions for path r_1^1 and r_2^1 at the *equilibrium* state are respectively 0.667 and 0.333. Due to the symmetry in this scenario, source s_2 behaves in exactly the same manner, and achieves the same equilibrium flow proportions for its two paths r_1^2 and r_2^2. Similarly, if we employ the *ebr* strategy, both sources will also gradually back off from the paths with the shared bottleneck link and arrive at an equilibrium state, where only 0.356 proportion of the flows of each source are routed through the path with the shared bottleneck link.

Now consider the scenario where after the above equilibrium state is achieved, the offered load at s_1 increases from 22 to 25 whereas the offered load at s_2 decreases from 22 to 15. Given the new load at both sources, routing flows along the paths using the old equilibrium proportions no longer results in an equilibrium state. In particular, source s_1 sees a blocking probability of $b_1^1 = 0.0784$ on path r_1^1 and a blocking probability of $b_2^1 = 0.0216$ on path r_2^1. On the other hand, source s_2 sees a blocking probability of $b_1^2 = 0.0018$ on path r_1^2 and a blocking probability of $b_2^2 = 0.0216$ on path r_2^2. Hence, in an effort to equalize the blocking probabilities on both paths, s_1 will direct more flows to path r_2^1 and s_2 will direct more flows to path r_1^2. The new adaptation process is shown on the right side (scenario II, starting with iteration 10) of Figure 4(a). From the figure we see that as source s_1 directs more flows to path r_2^1, the observed blocking probability on path r_2^1 gradually increases while the observed blocking probability on path r_1^1 gradually decreases. These two blocking probabilities are eventually equalized at around 0.022. The proportions of flows routed along the two paths by source s_1 during this adaptation process are shown in Figure 4(b), where the equilibrium flow proportions for paths r_1^1 and r_2^1 are around 0.537 and 0.463, respectively. The convergence process for source s_2 is similar, where more flows are routed along path r_1^2, eventually resulting in both of its two paths having an observed blocking probability of around 0.022.

It is interesting to note that each source adapts to the load changes *not* with any *global* objective *but* with a *local* objective of equalizing blocking probabilities or rates among all paths to a given destination. This in turn results in an overall near-optimal stable system performance. For example, in scenario I, both source s_1 and source s_2 have

an equal capacity share on the bottleneck link $4 \rightarrow 6$, each with a virtual capacity of 12. But as the load changes at each source, source s_1 starts routing more flows to path r_2^1, whereas source s_2 starts backing off from the path r_2^2, thereby allowing s_1 to grab more capacity share on the bottleneck link. The changes in the virtual capacity of the shared link seen by each source are shown in Figure 4(c). At the end, source s_1 has a virtual capacity of 18 from the shared bottleneck link, while source s_3 has a virtual capacity of 6. Due to this change in capacity shares, the blocking probability observed by source s_1 is reduced from 0.0595 at the onset of load change to 0.0225 in the end while that of s_2 goes up from 0.0084 to 0.0202. However, as a consequence of these self-adaptations at the two sources, the overall *system* blocking probability is reduced from 0.0404 to 0.022 (Figure 4(a)).

2.3 Self-Refrained Alternative Routing

In the virtual capacity model, all paths between a source and a destination are treated equally. Since an admitted flow consumes bandwidth and buffer resources at all the links along a path, clearly path length is also an important factor that we must take into consideration. There is a fundamental trade-off between minimizing the resource usage by choosing shorter paths and balancing the network load by using lightly loaded longer paths. As a general principle, it is preferable to route a flow along *minhop* (i.e. shortest) paths than paths of longer length (also referred to as *alternative* paths)[4]. By preferring minhop paths and discriminating against alternative paths, we not only reduce the overall resource usage but also limit the so-called "knock-on" effect [13, 14], thereby ensuring the stability of the whole system.

The "knock-on" effect refers to the phenomenon where using alternative paths by some sources forces other sources whose minhop paths share links with these alternative paths to also use alternative paths. This cascading effect can cause a drastic reduction in the overall throughput of the network. In order to deal with the "knock-on" effect, trunk reservation [14] is employed where a certain amount of bandwidth on a link is reserved for minhop paths only. With trunk reservation, a flow may be rejected even if sufficient resources are available to accommodate it. A flow along a path longer than its minhop path is admitted only if the available bandwidth even after admitting this flow is greater than the amount of trunk reserved. Trunk reservation provides a simple and yet effective mechanism to control the "knock-on" effect. However, it requires that core routers figure out whether a setup request for a flow is sent along its minhop path or not. This certainly introduces undesirable burden on core routers. To avoid this, we propose a *self-refrained* alternative routing method, which when employed at a source provides an adaptive way to discriminate against "bad" alternative paths *without explicit trunk reservation*.

Consider a source-destination pair. Suppose there are k^{min} number of minhop paths between this source-destination pair, and let R^{min} denote the set of these minhop paths. The set of alternative paths is denoted by R^{alt}. Thus the set of all candidate paths $R = R^{min} \cup R^{alt}$. The basic idea behind the *self-refrained* alternative routing method is to ensure that *an alternative path is used to route flows between the source-destination pair only if it has a "better quality" (measured in flow blocking probability) than any of the minhop paths.* Formally, for a path

[4]Although the virtual capacity model does not explicitly take path length into account, it does tend to discriminate against longer paths implicitly, as longer paths are likely to have a higher blocking probability in practice.

1. PROCEDURE VCR()
2. Set mean blocking rate of minhop paths, $\bar{\beta}^{(n)} = \frac{\sum_{r \in R^{min}} \alpha_r^{(n)} b_r^{(n)}}{k^{min}}$
3. Set minimum of minhop path's blocking probability, $b^* = \min_{r \in R^{min}} b_r^{(n)}$
4. For each path $r \in R$
5. Compute virtual capacity $vc_r^{(n)} = E_{vc}^{-1}(\nu_r^{(n)}, b_r^{(n)})$
6. For each minhop path $r \in R^{min}$
7. Compute target load ν_r^* such that $\bar{\beta}^{(n)} = \nu_r^* E(\nu_r^*, vc_r^{(n)})$
8. For each alternative path $r \in R^{alt}$
9. Compute target load ν_r^* such that $\psi b^* = E(\nu_r^*, vc_r^{(n)})$
10. For each path $r \in R$
11. Compute new proportion $\alpha_r^{(n+1)} = \frac{\nu_r^*}{\sum_{r \in R} \nu_r^*}$
12. END PROCEDURE

Figure 5: The *vcr* procedure

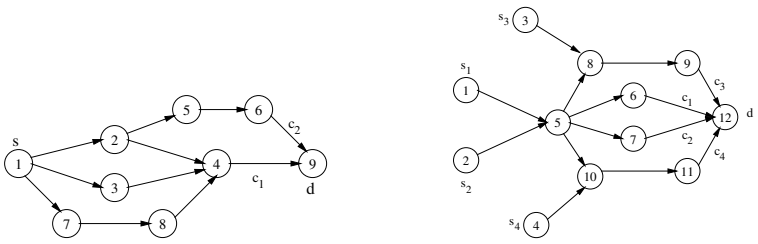
$r \in R^{min}$, let b_r denote the observed flow blocking probability on path r. The minimum flow blocking probability of all the minhop paths, $b^* = \min_{r \in R^{min}} b_r$, is used as the reference in deciding a target flow blocking probability for alternative paths. The target flow blocking for alternative paths is set to ψb^*, where ψ is a configurable parameter to limit the "knock-on" effect under system overloads. An alternative path $r' \in R^{alt}$ is selected to route flows only if it can *attain* the target flow blocking probability. In other words, its observed flow blocking probability $b_{r'}$ is less than or equal to ψb^*.

This *self-refrained* alternative routing method has several attractive features. By using b^* as the reference in determining a target flow blocking probability for alternative paths, it dynamically controls the extent of alternative routing according to both the load at the source and the overall system load. For example, if both the load at the source and the overall system load is light, the use of alternative paths will be kept at a minimum. However, if the load at the source is heavy but the overall system load is light, more alternative routes will be used by the source. Furthermore, by using only those alternative paths whose observed blocking probabilities are at most as high as the minimum of those of the minhop paths, we guarantee that the minhop paths are preferred to alternative paths. In particular, if an alternative path of a source-destination shares a bottleneck with one of its minhop paths, this alternative path is automatically "pruned". In addition, a source would gradually back off from an alternative path once its observed flow blocking probability starts increasing, thereby adapting gracefully to the change in the network load.

2.4 Virtual Capacity based Proportional Routing

By incorporating this *self-refrained* alternative routing method into the virtual capacity model, we devise a theoretical adaptive proportional routing scheme, which is referred to as the *Virtual Capacity based Routing* (vcr) scheme. In this *vcr* scheme, we use the *ebr* strategy[5] to proportion flows along the minhop paths, whereas proportions of flows

[5]We adopted the *ebr* strategy as it is found to be more amenable for implementation.



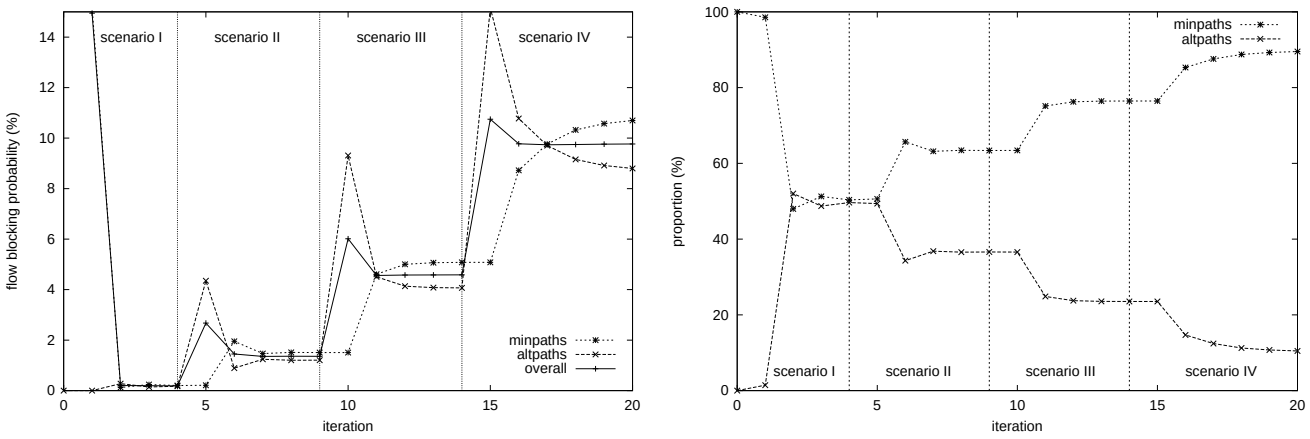
(a) *duck* (b) *fish*

Figure 6: Topologies used for illustration

along the alternative paths are computed using the target flow blocking probability ψb^*, as in the *self-refrained* alternative routing method. The scheme is shown in Figure 5. Suppose the total load for a source-destination pair is ν. At a given step $n \geq 0$, let $\nu_r^{(n)} = \alpha_r^{(n)}\nu$ be the amount of the load currently routed along a path $r \in R$, and let $b_r^{(n)}$ be its observed blocking probability on the path. Then the virtual capacity of path r is given by $vc_r = E_{vc}^{-1}(\nu_r^{(n)}, b_r^{(n)})$ (line 5). For each minhop path, the mean blocking rate of all the minhop paths, $\bar{\beta}^{(n)}$, is used to compute a new target load (lines 6-7). Similarly, for each alternative path, a new target load is determined using the target blocking probability ψb^* (lines 8-9). Given these new target loads for all the paths, the new proportion of flows, $\alpha_r^{(n+1)}$, for each path r is obtained in lines 10-11, resulting in a new load $\nu_r^{(n+1)} = \alpha_r^{(n+1)}\nu$ on path r.

In the following we illustrate through numerical examples how the *vcr* scheme uses alternative paths in a judicious and self-adaptive manner. First consider the *duck* topology shown in Figure 6(a). Let r_1^{min} and r_2^{min} denote, respectively, the two minhop paths $1 \to 2 \to 4 \to 9$ and $1 \to 3 \to 4 \to 9$. Similarly let r_3^{alt}, and r_4^{alt} denote, respectively, the two alternative paths $1 \to 2 \to 5 \to 6 \to 9$ and $1 \to 7 \to 8 \to 4 \to 9$. The two minhops r_1^{min} and r_2^{min} share the bottleneck link $4 \to 9$ with the alternative path r_4^{alt}. On the other hand, the minhop path r_1^{min} and the alternative path r_3^{alt} share the link $1 \to 2$, which is *not* a bottleneck link. The capacities c_1 and c_2 of bottleneck links are set to 20. Assume that a load of 30 is offered at source s. With the ψ parameter set to 0.8 and starting with a set of arbitrary proportions for the four paths, the *vcr* scheme would eventually reach a set of equilibrium proportions, which are $\alpha_1^{min} = 0.255$, $\alpha_2^{min} = 0.255$, $\alpha_3^{alt} = 0.490$ and $\alpha_4^{alt} = 0.000$, respectively. We see that a total 51% of the flows are routed through the bottleneck link $4 \to 9$. This link is shared equally by the two minhop paths, r_1^{min} and r_2^{min}, each with a blocking probability of 0.0508. The alternative path, r_3^{alt}, which also share the bottleneck link with the two minhop paths, is effectively cut off from the link and not used at all. This is because routing any flows through r_3^{alt} would only increase the resource usage without resulting in any decrease in the overall blocking probability. In contrast, the alternative path, r_3^1, is used to route 47% of the flows, with a blocking probability of 0.0406, which matches the target blocking probability for the alternative paths, $\psi b^* = 0.0406$. Since r_3^{alt} shares a *non-bottleneck* link with r_1^{min}, routing flows through r_3^{alt} helps reduce the overall blocking probability.

In the next example, we demonstrate how the *vcr* scheme controls the extent of alternative routing to adapt to the changes in traffic load. Consider the *fish* topology shown in Figure 6(b). The nodes 1, 2, 3, and 4 are the source



(a) *convergence process*

(b) *adaptation of proportions*

Figure 7: Illustration of usage of alternative paths in *vcr*

nodes and node 12 is the destination node. The nodes 1 and 2 each have two minhop paths and two alternative paths to the destination node 12. Other two source nodes, 3 and 4, have just one minhop path to the destination node 12. The alternative paths of source nodes 1 and 2 share the bottleneck links $9 \rightarrow 12$ and $11 \rightarrow 12$ with the minhop paths of 3 and 4. Assume that the capacities c_1, c_2, c_3 and c_4 of the bottleneck links are all set to 20. We consider four scenarios where the offered load at source nodes 1 and 2 are fixed at 20 while the offered load at source nodes 3 and 4 are increased from 0 to 5, 10 and 15 in scenarios I, II, III, and IV respectively, and study how source nodes 1 and 2 adjust their flow proportions on the alternative paths. Figures 7(a) and 7(b) show, from the perspective of source node 1, the adaptation process as reflected in the flow blocking probabilities and proportions associated with the minhop paths and the alternative paths. Note that due to the symmetry, source node 2 behaves in exactly the same manner. Hence we will focus only on the behavior of source node 1.

Suppose initially both source nodes 1 and 2 use only their minhop paths. This results in a high blocking probability of 0.1588 on the minhop paths. As both source nodes sense the availability of the alternative paths and start routing flows through them, the blocking probability on the minhop paths drops quickly, resulting in an overall blocking probability of around 0.0019 (see Scenario I in Figure7(a)). At the equilibrium state, the total proportion of flows routed along the two alternative paths is 0.4964 (see Scenario I in Figure7(b)). When sources nodes 3 and 4 become active with a load of 5 each, the blocking probability on the two alternative paths shoots up to 0.0435 from 0.0017. The source 1 reacts to this by reducing the proportion of the flows routed to the alternative paths from 0.4964 to 0.3659, pulling the overall blocking probability down to 0.0136 (see Scenario II in Figure7(a) and Figure7(b)). Note that at the equilibrium state, the blocking probabilities of the alternative paths are kept at 0.8 times of that of the minhop paths, as determined by the parameter $\psi = 0.8$. As the load at source nodes 3 and 4 increases further from 5 to 10, then to 15, both source node 1 and source node 2 keep backing off from their alternative paths to yield more

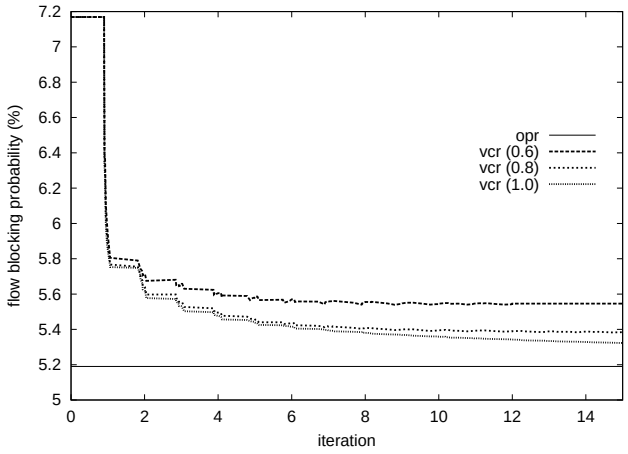


Figure 8: Comparison of *vcr* and *opr*

capacity share to the minhop paths of source nodes 3 and 4 (see Scenarios III and IV in Figure7(a) and Figure7(b)). At the end, the proportion of flows routed by source node 1 to the alternative paths is eventually decreased to only 0.1047, yielding an overall blocking probability of 0.0977. This example shows that the *vcr* scheme can adaptively respond to the traffic load changes along the alternative paths by adjusting the proportion of flows routed along these paths. It was argued [37] that selection of maximally disjoint paths yields better blocking performance. The above results show that, using the virtual capacity model and *self-refrained* alternative routing method, the *vcr* scheme judiciously proportions traffic among minhop and alternative paths *without actually being aware of where the shared bottleneck links are*.

We now illustrate the convergence of *vcr* using a larger *isp* topology shown in Figure 10(a). The topology and traffic characteristics are described in Section 4.1. For this study, a load of 0.35 is offered uniformly between all the border nodes. Figure 8 shows the overall blocking probability as a function of the iteration number. The performance of *vcr* is shown for three different values of ψ: $0.6, 0.8, 1.0$. The blocking probability corresponding to optimal proportions computed by *opr* is also shown for reference. First thing to note is that *vcr* converges for all values of ψ. Furthermore, it approaches the convergence point within 10 iterations. The performance of *vcr* with $\psi = 0.8$ is slightly better than with $\psi = 0.6$ while not much difference between ψ values 0.8 and 1.0. This difference between ψ values 0.6 and 0.8 indicates the blocking performance penalty paid for being extra cautious in avoiding knock-on effect. Note that longer alternate paths are naturally discriminated because they are likely to have higher blocking probability. The parameter ψ provides additional safe guarding against knock-on effect. Our results show that 0.8 is a reasonable setting for ψ. Finally, it can be seen that the blocking probability achieved by *vcr* based proportioning is within 0.2% of the optimal blocking probability. These results affirm that *vcr* yields near-optimal proportions.

```
1.  PROCEDURE PSR-ROUTE()
2.    Select an eligible path r = wrrps(R^elg)
3.    Increment flow counter, n_r = n_r + 1
4.    If failed to setup connection along r
5.      Decrement failure counter, f_r = f_r − 1
6.      If failures reached limit, f_r == 0
7.        Remove r from eligible set, R^elg = R^elg − r
8.        If eligible set is empty, R^elg == ∅
9.          Reset eligible set, R^elg = R
10.         For each path r ∈ R
11.           Reset failure counter, f_r = γ_r
12. END PROCEDURE
```

(a) proportional routing

```
1.  PROCEDURE PSR-PROPO-COMPU()
2.    For each path r ∈ R
3.      Compute blocking probability, b_r = ηγ_r / n_r
4.      Assign a proportion, α_r = n_r / Σ_{r̃∈R} n_r̃
5.    Set target blocking probability, b* = min_{r∈R^min} b_r
6.    For each alternative path r' ∈ R^alt
7.      If blocking probability high, b_r' >= b*
8.        Decrement failure limit, γ_r' = γ_r' − 1
9.      If blocking probability low, b_r' < ψb*
10.       Increment failure limit, γ_r' = γ_r' + 1
11. END PROCEDURE
```

(b) computation of proportions

Figure 9: The *psr* procedure

3 Proportional Sticky Routing: A Practical Implementation of VCR

In the previous section, we presented an analytical framework for modeling adaptive proportional routing. In particular, based on this framework we described a theoretical adaptive routing scheme — the *vcr* scheme, and demonstrated its self-adaptivity through several numerical examples. There are two difficulties involved in implementing the virtual capacity model. First, computation of virtual capacity and target load using Erlang's Loss Formula can be quite cumbersome. Second, and perhaps more importantly, the accuracy in using Erlang's Loss Formula to compute virtual capacity and new load relies critically on steady-state observation of flow blocking probability. Hence small statistic variations may lead to erroneous flow proportioning, causing undesirable load fluctuations. In order to circumvent these difficulties, we are interested in a simple yet robust implementation of the *vcr* scheme. In this section we present such an implementation which we refer to as the *proportional sticky routing* (psr) scheme[6].

The *psr* scheme can be viewed to operate in two stages: 1) proportional flow routing, and 2) computation of flow proportions. The proportional flow routing stage proceeds in *cycles* of variable length. During each cycle incoming flows are routed along paths selected from a set of eligible paths. A path is selected with a frequency determined by a prescribed proportion. A number of cycles form an *observation period*, at the end of which a new flow proportion for each path is computed based on its observed blocking probability. This is the computation of flow proportion stage. As in the *vcr* scheme, flow proportions for minhop paths of a source-destination pair are determined using the *ebr* strategy, whereas flow proportions for alternative paths are determined using a target blocking probability. In the following we will describe these two stages in more detail.

[6]The *psr* scheme essentially does proportional routing while obtaining proportions through a form of sticky routing.

Proportional flow routing

Given an arbitrary source-destination pair, let R be the set of candidate paths between the source-destination pair, where $R = R^{min} \cup R^{alt}$. We associate with each path $r \in R$, a *maximum permissible flow blocking* number γ_r and a corresponding *flow blocking counter* f_r. For each minhop path $r \in R^{min}$, $\gamma_r = \hat{\gamma}$, where $\hat{\gamma}$ is a configurable system parameter. For each alternative path $r' \in R^{alt}$, the value of $\gamma_{r'}$ is dynamically adjusted between 1 and $\hat{\gamma}$, as will be explained later. As shown in Figure 9(a), at the beginning of each cycle, f_r is set to γ_r. Every time a flow routed along path r is blocked, f_r is decremented. When f_r reaches zero, path r is considered *ineligible*. At any time only the set of *eligible* paths, denoted by R^{elg}, is used to route flows. A path from current eligible path set R^{elg} is selected using a weighted-round-robin-like path selector (*wrrps*). The *wrrps* procedure is described in the Appendix. Once R^{elg} becomes empty, the current cycle is ended and a new cycle is started with $R^{elg} = R$ and $f_r = \gamma_r$.

Computation of flow proportions

Flow proportions $\{\alpha_r, r \in R\}$ are recomputed at the end of each observation period (see Figure 9(b)). An observation period consists of η cycles, where η is a configurable system parameter used to control the robustness and stability of flow statistics measurement. During each observation period, we keep track of the number of flows routed along each path $r \in R$ using a counter n_r. At the beginning of an observation period, n_r is set to 0. Every time path r is used to route a flow, n_r is incremented. Since an observation period consists of η cycles, and in every cycle, each path r has exactly γ_r flows blocked, the observed flow blocking probability on path r is $b_r = \frac{\eta\gamma_r}{n_r}$. For each minhop path $r \in R^{min}$, its new proportion α_r is recomputed at the end of an observation period and is given by $\alpha_r = n_r/n_{total}$, where $n_{total} = \sum_{r \in R} n_r$ is the total number of flows routed during an observation period. Recall that for a minhop path $r \in R^{min}$, $\gamma_r = \hat{\gamma}$. Hence $\alpha_r b_r = \frac{n_r}{n_{total}} \frac{\eta\gamma_r}{n_r} = \frac{n_r}{n_{total}} \frac{\eta\hat{\gamma}}{n_r} = \frac{\eta\hat{\gamma}}{n_{total}}$. This shows that the above method of assigning flow proportions for the minhop paths equalizes their flow blocking rates.

As in the *vcr* scheme, we use the minimum blocking probability among the minhop paths, $b^* = \min_{r \in R^{min}} b_r$, as the reference to control flow proportions for the alternative paths. This is done implicitly by dynamically adjusting the maximum permissible flow blocking parameter $\gamma_{r'}$ for each alternative path $r' \in R^{alt}$. At the end of an observation period, let $b_{r'} = \frac{\eta\gamma_{r'}}{n_{r'}}$ be the observed flow blocking probability for an alternative path r'. If $b_{r'} > b^*$, $\gamma_{r'} := \max\{\gamma_{r'} - 1, 1\}$. If $b_{r'} < \psi b^*$, $\gamma_{r'} := \min\{\gamma_{r'} + 1, \hat{\gamma}\}$. If $\psi b^* \leq b_{r'} \leq b^*$, $\gamma_{r'}$ is not changed. By having $\gamma_{r'} \geq 1$, we ensure that some flows are occasionally routed along alternative path r' to *probe* its "quality", whereas by keeping $\gamma_{r'}$ always below $\hat{\gamma}$, we guarantee that minhop paths are always preferred to alternative paths in routing flows. The new proportion for each alternative path r' is again given by $\alpha_{r'} = n_{r'}/n_{total}$. Note that since $\gamma_{r'}$ is adjusted for the next observation period, the *actual* number of flows routed along alternative path r' will be also adjusted accordingly.

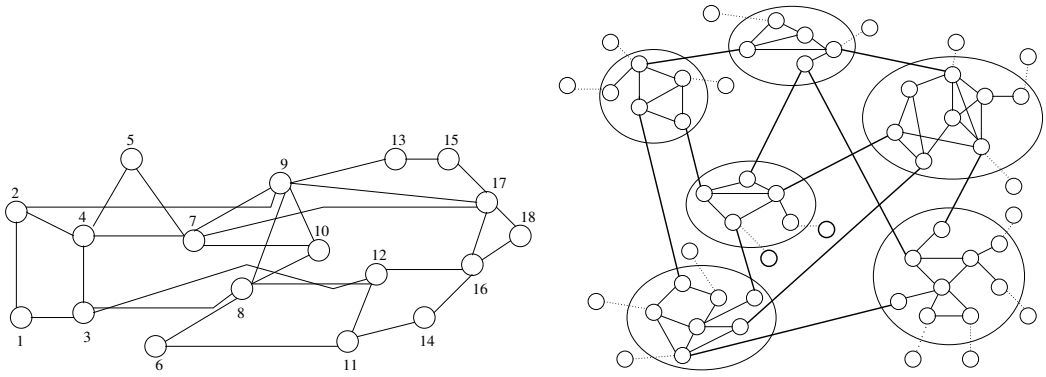
Table 1: Comparison of proportioning in *vcr* and *psr*

Topo	Scenario		*vcr*	*psr*
kite	$\nu_{s_1} = 22, \nu_{s_2} = 22$	$\alpha_{1\to4\to6}$	0.356	0.351
		$\alpha_{2\to4\to6}$	0.356	0.357
kite	$\nu_{s_1} = 25, \nu_{s_2} = 15$	$\alpha_{1\to4\to6}$	0.447	0.455
		$\alpha_{2\to4\to6}$	0.208	0.193
duck	$\nu_s = 30$	$\alpha_{1\to2\to4\to9}$	0.255	0.269
		$\alpha_{1\to3\to4\to9}$	0.255	0.236
		$\alpha_{1\to2\to5\to6\to9}$	0.490	0.474
		$\alpha_{1\to7\to8\to4\to9}$	0.000	0.021

Comparison with *vcr*

The *psr* scheme preserves the self-adaptivity of the theoretical *vcr* scheme by controlling the number of flows routed along a path r in each cycle using γ_r and by re-adjusting flow proportions after every observation period. For example, if the load along a path r increases, causing the number of flows blocked to quickly reach γ_r, the source will automatically back off from this path by eliminating it from the eligible path set for the rest of the cycle. If this situation persists, at the end of the observation period, the new flow proportion for path r will be reduced. Likewise, if the load on path r decreases, its new flow proportion will be increased at the end of the observation period. This is particularly true for alternative paths with their dynamically adjusted γ_r. Furthermore, because the length of each cycle is not fixed but determined by how fast each eligible path reaches its maximal permissible blocks, the length of an observation period also varies. This *self-adjusting* observation period allows the *psr* scheme to respond to the system load fluctuations in an elastic manner. If the system load changes suddenly, the old flow proportions would result in rapid termination of cycles, which would in turn lead to faster conclusion of the current observation period. New flow proportions will thus be re-computed to adapt to the system load. On the other hand, if the system load is stable, the observation periods will also be stabilized, with increasingly accurate calibration of the flow proportions. As a result, flow proportioning will eventually converge to the equilibrium state.

Table 1 compares the simulation results obtained using the *psr* scheme with the corresponding numerical results obtained using the theoretical *vcr* scheme under various settings. The capacities of all bottleneck links are set to 20. The observation period η in *psr* is set to 3 cycles to average out the random effects before recomputing proportions. The trunk reservation parameter ψ is set to 0.8 and the maximum permissible flow blocking parameter, $\hat{\gamma}$ is set to 5. The table shows the proportions assigned to each path under each setting. In all the settings, the difference in proportions between these two schemes is not significant. An interesting case shown in the table is that of *duck* topology where *vcr* assigns zero proportion to the alternative path $1 \to 7 \to 8 \to 4 \to 9$ since it shares a bottleneck link $(4 \to 9)$ with minhop paths. The *psr* scheme routes 0.021 proportion of flows to this path. This is because *psr* has to route some flows to a path to probe its quality. However, note that this is a small proportion and doesn't severely affect the performance. These results show that the *psr* scheme closely approximates the *vcr* scheme.



(a) *isp* (b) *rand*

Figure 10: Topologies used in performance evaluation

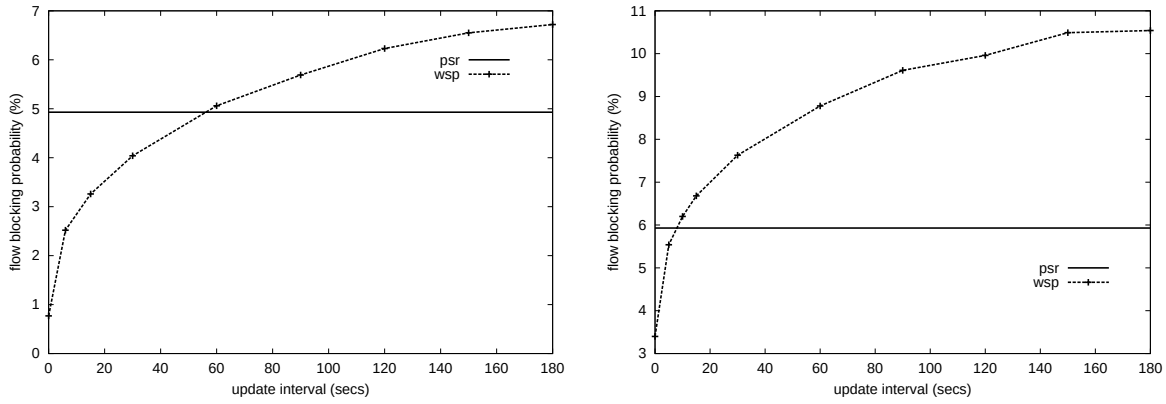
4 Performance Evaluation and Analysis

In this section, we evaluate the performance of the proposed localized QoS routing scheme *psr* and compare it with the global QoS routing scheme *widest shortest path* (*wsp*). We first describe the simulation environment and then compare the performance of *psr* and *wsp* in terms of the overall blocking probability, routing stability and overhead.

4.1 Simulation Environment

Figure 10 shows the two topologies, *isp* and *rand*, used in our study. The *isp* topology of an ISP backbone network is also used in [1, 20]. The *rand* topology is a random graph generated by GT-ITM [39] and used in [40]. For simplicity, all the links are assumed to be bidirectional and of equal capacity in each direction. The *rand* topology has three types of links: *thin*, *thick* and *dotted* while *isp* topology has only *thin* links. All *thin* links have same capacity with C_1 units of bandwidth and similarly all the *thick* links have C_2 units. The *dotted* links are the access links and for the purpose of our study their capacity is assumed to be infinite. Flows arriving into the network are assumed to require one unit of bandwidth. Hence a link with capacity C can accommodate at most C flows simultaneously.

The dynamics of flows in the network is modeled as follows (similar to the model used in [34]). A set of nodes in the network is designated as capable of being source/destination nodes of flows. In case of *rand* topology, only the nodes attached to the *dotted* access links are assumed to be end points of flows. In case of *isp* topology, we consider two settings. In the first setting, all nodes are included in this set and in the second setting, only the 9 border nodes, namely $1, 2, 5, 6, 11, 13, 14, 15, 18$ are included. Flows arrive at a source node according to a Poisson process with rate λ. The destination node of a flow is chosen randomly from the designated set of nodes except the source node. The holding time of a flow is exponentially distributed with mean $1/\mu$. Following [34], the offered network load on *isp* is given by $\rho = \lambda N\bar{h}/\mu L_1 C_1$, where N is the number of source nodes, L_1 the number of links, and $\bar{h}$ is the mean number of hops per flow, averaged across all source-destination pairs. Similarly the offered load on *rand* is



(a) *isp* ($\rho = 0.60$)

(b) *rand* ($\rho = 0.40$)

Figure 11: Impact of update interval

given by $\rho = \lambda N \bar{h}/\mu(L_1 C_1 + L_2 C_2)$, where L_1 and L_2 are the number of *thin* and *thick* links respectively. The parameters used in our simulations are $C_1 = 20$, $C_2 = 40$, $1/\mu = 60$ sec. The topology specific parameters are $N = 18$, $L_1 = 60$, $\bar{h} = 2.36$ for *isp* and $N = 56$, $L_1 = 100$, $L_2 = 22$, $\bar{h} = 4.38$ for *rand*. The average arrival rate at a source node λ is set depending upon the desired load.

The parameters in the simulation are set as follows by default. Any change from these settings is explicitly mentioned wherever necessary. The values for configurable parameters in *psr* are $\eta = 3$, $\hat{\gamma} = 5$, and $\psi = 0.8$. For each source-destination pair, all the paths between them whose length is at most one hop more than the minimum number of hops are chosen as the candidate paths. The average number of candidate (*minhop* and *minhop+1*) paths used in *psr* are $5.16(1.39 + 3.77)$ in *rand*, and $4.63(1.50 + 3.13)$ and $5.20(1.53 + 3.67)$ respectively in the first and the second settings of *isp*. Each run simulates arrival of $1,000,000$ flows and the results corresponding to the latter half of the simulation are reported here.

4.2 Blocking Probability

The performance of *wsp* and *psr* is compared by measuring the blocking probability under various settings. We first present the impact of update interval on the performance of *wsp* and show how the blocking probability increases rapidly as update interval is increased. We then demonstrate the adaptivity of *psr* by varying the overall load. Finally we compare the performance of these two schemes under non-uniform load conditions and show that *psr* is better at alleviating the effect of "hot spots".

Varying update interval

Figure 11 compares the performance of *wsp* and *psr* for both *isp* and *rand* topologies. The offered load was set to 0.60 in case of *isp* and 0.40 in case of *rand*. The performance is measured in terms of the overall flow blocking

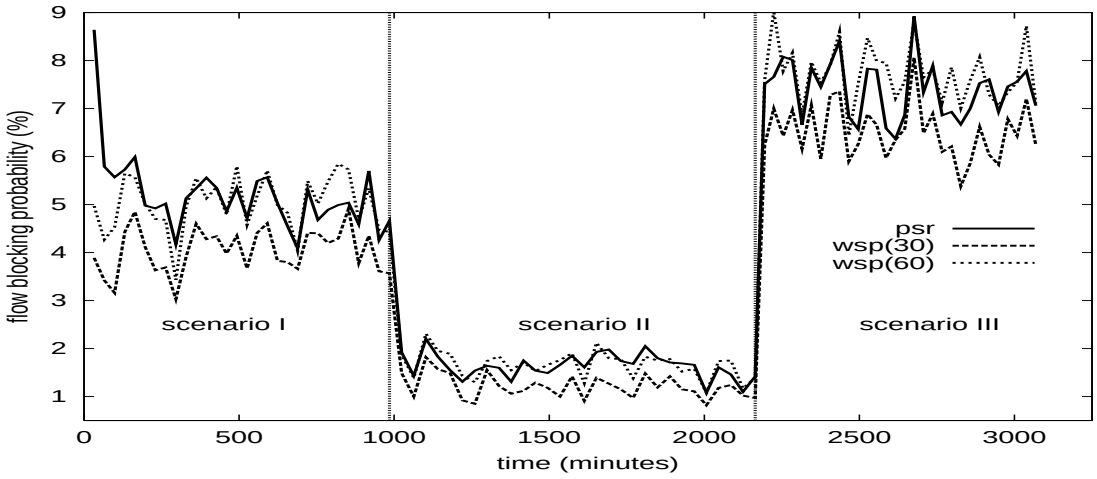


Figure 12: Performance under varying load

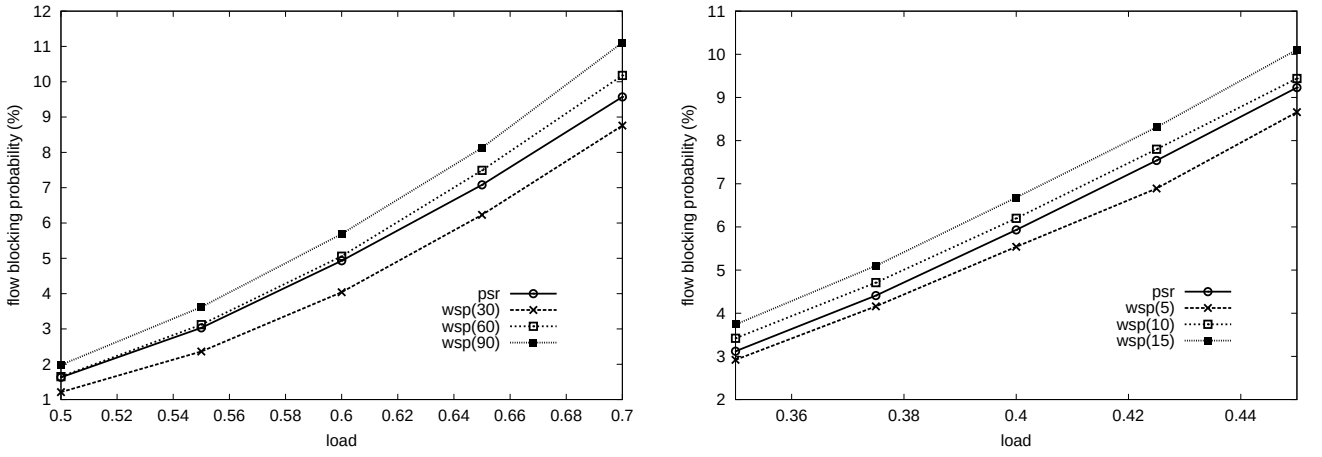
probability, which is defined as the ratio of the total number of blocks to the total number of flow arrivals. The overall blocking probability is plotted as a function of the update interval used in *wsp* for periodic updates[7]. From the figures, we see that as the update interval of *wsp* increases, the blocking probability of *wsp* rapidly approaches that of *psr* and is worse for larger update intervals. In the case of *isp* topology, *psr* performs better than *wsp* when the update interval is greater than 60 sec. For *rand* topology, this crossover happens at a much smaller update interval of less than 10 sec. The reasons for this poor performance of *wsp* are further investigated and explained later in this section.

Varying offered load

We now illustrate the adaptivity of *psr* by varying the offered load. We initially offer a load of 0.60 as was done in the earlier simulation and then this overall load is decreased to 0.50 and again increased to 0.65. We plot the blocking probability under *psr* and *wsp* as a function of time in Figure 12. The performance of *wsp* is shown for two update intervals: 30 sec and 60 sec. Starting with arbitrary initial proportions, *psr* quickly converges and performs as well as *wsp*(60). When the load is decreased, *psr* adapts to the change and maintains its relative performance. Finally, when the load is increased to 0.65, once again it reacts promptly and performs slightly better than *wsp*(60). This leads us to study how the amount of load affects the relative performance of these schemes.

Figure 13 shows the blocking performance of these two schemes as a function of the offered network load. As before, the performance is measured in terms of the overall flow blocking probability. The network load is varied from 0.50 to 0.70 in case *isp* and 0.35 to 0.45 in case of *rand*. The performance of *wsp* is plotted for three update intervals of 30, 60 and 90 for the *isp* case and similarly for 5, 10, and 15 in case of *rand*. It is clear that *psr* performs as well as *wsp*(60) at low loads and better at high loads. In case of *rand*, *psr*'s performance is better than *wsp* with

[7]Note that blocking performance of *wsp* with threshold triggered updates with hold-down timer T would be no better than periodic updates with update interval T. The difference is in the amount of update message overhead.



(a) *isp* (b) *rand*

Figure 13: Performance under various loads

an update interval of 10 sec. The poor performance of *wsp* particularly in case of *rand* is investigated and explained below. So far, we assumed the first setting of *isp* where any node can potentially be a source or a destination, while in *rand*, only a few nodes are considered to be end points of flows. This causes the traffic distribution across the network to be more unbalanced in *rand* than in *isp* and *wsp* performs poorly under such a setting. To illustrate this, we have compared the performance of these schemes under *isp* with non-uniform load conditions and the results are described below.

Varying non-uniform traffic

It is likely that a source node receives a larger number of flows to a few specific destinations [6], i.e, a few destinations are "hot". Ideally a source would like to have more up-to-date view of the QoS state of the links along the paths to these "hot" destinations. In the case of *wsp*, this requires more frequent QoS state updates, resulting in increased overhead. But in the case of *psr*, because of its adaptivity and statistics collection mechanism, a source does have more accurate information about the frequently used routes and thus alleviates the effect of "hot spots". We illustrate this by introducing increased levels of traffic between certain pairs of network nodes ("hot pairs"), as was done in [1]. Apart from the normal load that is distributed between all source-destination pairs, an additional load (hot load) is distributed among all the hot pair nodes. The hot pairs chosen for *isp* topology are $(2, 16)$, $(3, 17)$, and $(9, 11)$.

We consider three scenarios under *isp*. In scenario I, a load of 0.50 is offered uniformly among all the nodes as was done in earlier simulations. In scenario II, an additional load of 0.05 is offered between hot pairs only and in scenario III this additional load is further increased to 0.10. Figure 14 shows the blocking performance of the two schemes under different scenarios as a function of time. Under scenario I, starting with arbitrary initial proportions, *psr* quickly converges to a stable state where its blocking probability is similar to that of *wsp*(60). But in scenario

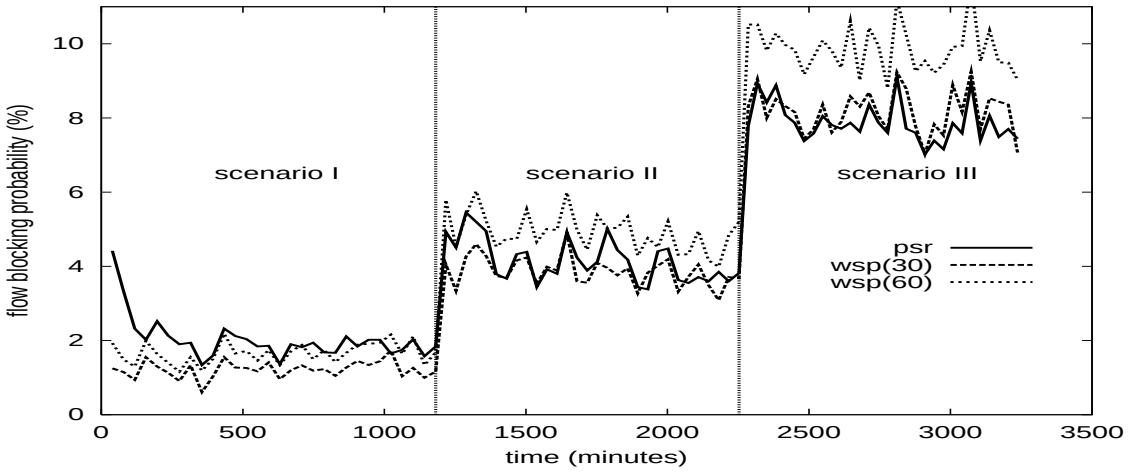
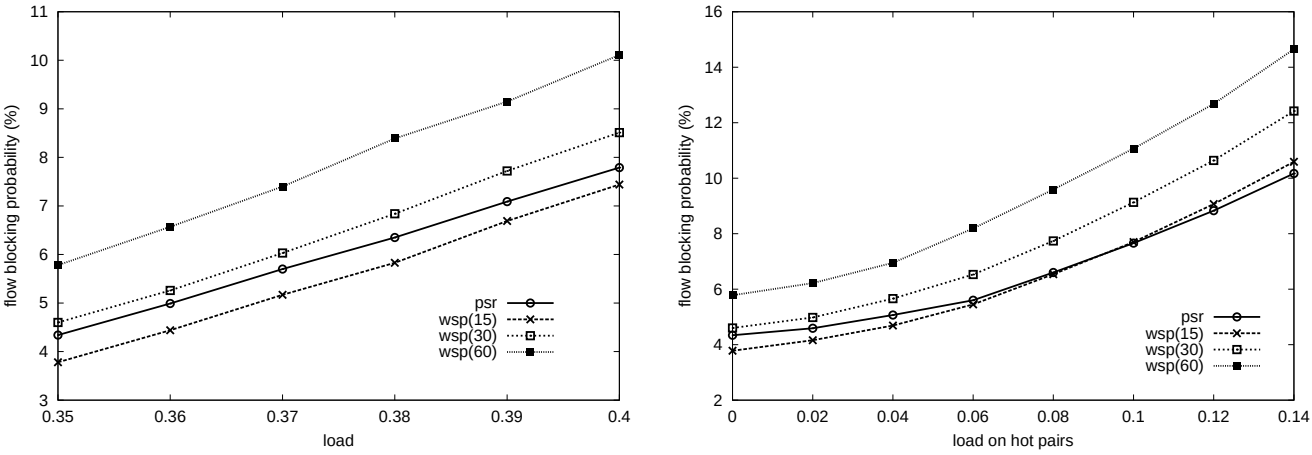


Figure 14: Performance under varying non-uniform load



(a) *fewer sources* (b) *hot pairs*

Figure 15: Performance under various non-uniform load conditions

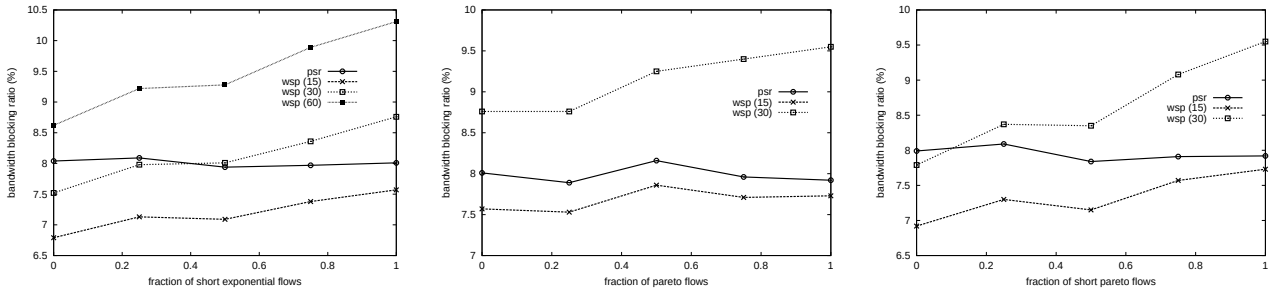
II with additional load between hot pairs, *psr* approaches the performance of *wsp*(30) and even better in scenario III where the load between hot pairs is higher. These results illustrate the degradation in performance of *wsp* and improvement in relative performance of *psr* under non-uniform load conditions.

We have further investigated the impact of non-uniform load on the relative performance of these schemes by varying the amount of non-uniform load. First, we consider the second setting of *isp* where load is offered only between the border nodes. This is a reasonable setting since these edge nodes are likely to be ingress and egress nodes for flows passing through this domain. We ran the simulations varying the load on border nodes from 0.35 to 0.40. Figure 15(a) shows the results of these simulations. It can be seen that, across all loads, *psr* performs better than *wsp* with 30 sec update interval. We then fixed the load on border nodes at 0.35 and varied the additional load offered on hot pairs. Figure 15(b) shows the blocking performance of the schemes as a function of the additional load. When there is no additional load on hot pairs, performance of *psr* is similar to *wsp*(30). As the additional load on hot pairs increases, *psr* does progressively better in comparison to *wsp* and at hot load of 0.10 it performs as well as *wsp* with an update interval of 15 sec and even better at higher hot loads. This not only shows the limitation of global routing schemes such as *wsp* but also illustrates the self-adaptivity of localized proportional routing schemes such as *psr*.

4.3 Heterogeneous Traffic

The discussion so far is focussed on the case where the traffic is homogeneous, i.e., all flows request for one unit of bandwidth and their holding times are derived from the same exponential distribution with a fixed mean value. Here we study the applicability of *psr* in routing heterogeneous traffic where flows could request for varying bandwidths with their holding times derived from different distributions. We demonstrate that *psr* is insensitive to the duration of individual flows and hence we do not need to differentiate flows based on their holding times. We also show that when the link capacities are considerably larger than the average bandwidth request of flows, it may not be necessary to treat them differently and hence *psr* can be used *as is* to route heterogeneous traffic.

Consider the case of traffic with k types of flows, each flow of type i having a mean holding time $1/\mu_i$ and requesting bandwidth B_i. Let ρ_i be the offered load on the network due to flows of type i, where the total offered load, $\rho = \sum_{i=1}^{k} \rho_i$. The fraction of total traffic that is of type i, $\phi_i = \rho_i/\rho$. The arrival rate of type i flows at a source node, λ_i is given by $\lambda_i = \rho_i \mu_i LC/N\bar{h}B_i$, which is an extension of the formula presented in Section 4.1. To account for the heterogeneity of traffic, bandwidth blocking ratio is used as the performance metric for comparing different routing schemes. The bandwidth blocking ratio is defined as the ratio of the bandwidth usage corresponding to blocked flows and the total bandwidth usage of all the offered traffic. Suppose b_i is the observed blocking probability for flows of type i, then the bandwidth blocking ratio is given by $\frac{\sum_{i=1}^{k} \frac{b_i \lambda_i B_i}{\mu_i}}{\sum_{i=1}^{k} \frac{\lambda_i B_i}{\mu_i}}$. In the following, we compare the performance of *psr* and *wsp*, measured in terms of bandwidth blocking ratio, under different traffic conditions, varying the fractions ϕ_i to control the traffic mix.



(a) long and short exponential flows (b) exponential and pareto flows (c) long and short pareto flows

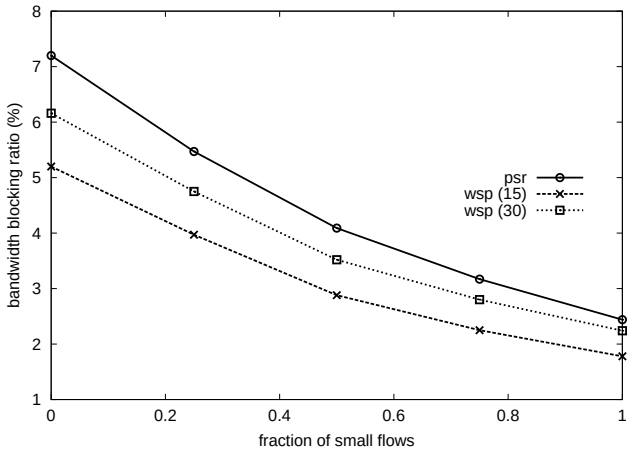
Figure 16: Performance under traffic with mixed holding times

Mixed holding times

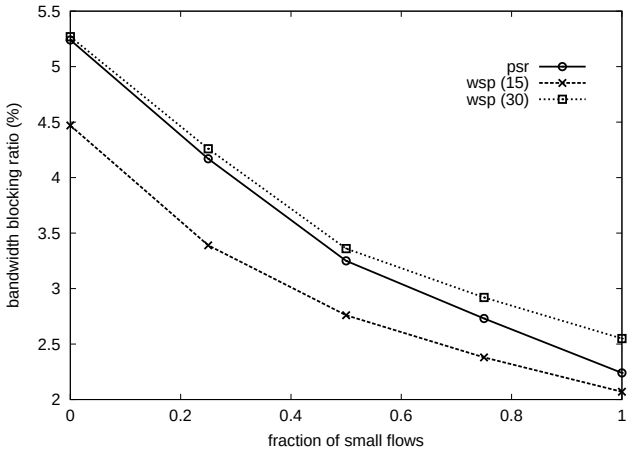
We now examine the case of traffic with 2 types of flows that request for the same amount of bandwidth, i.e, $B_1 = B_2 = 1$, but with different holding times. We consider three scenarios. In the first scenario, both types of flows have their holding times derived from exponential distribution but their means are different: 60 and 120 sec. In the second scenario, both types have the same mean holding time of 60 sec but their distributions are different: exponential and pareto. In the third scenario, holding times of both types of flows follow pareto distribution but their means are different: 60 and 120 sec. In all these scenarios, a load of 0.40 is offered between the border nodes in *isp*. Figure 16 shows the performance of *psr* and *wsp* under different scenarios.

Consider the first scenario where type 1 flows are *short* ($\frac{1}{\mu_1}$ = 60 sec) and type 2 flows are *long* ($\frac{1}{\mu_1}$ = 120 sec), but both are exponentially distributed. Figure 16(a) shows the bandwidth blocking ratio plotted as a function of the fraction ϕ_1 corresponding to short flows. It is quite evident that the performance of *wsp* degrades as the proportion of *short* flows increases while that of *psr* stays almost constant. The behavior of *wsp* is as expected since the shorter flows cause more fluctuation in the network QoS state and the information at a source node becomes more inaccurate as the QoS state update interval gets larger relative to flow dynamics. On the contrary, *psr* is insensitive to the duration of flows.

In the second scenario, a fraction of flows have their holding times derived from a pareto distribution while the rest have their holding times derived from an exponential distribution. The mean holding time of both the types is the same, 60 sec. The pareto distribution is heavy tailed with its tail controlled by a *shape* parameter. We have experimented with different shape values in the range 2.1 to 2.5 and found that results are similar. The results reported here correspond to a shape value of 2.2. In Figure 16(b), bandwidth blocking ratio is plotted as a function of the fraction of pareto type flows. As the fraction of pareto flows increases, the blocking under *wsp*(30) increases while it stays almost same under *wsp*(15). The number of short (much less than mean holding time) flows are more under the pareto distribution than the exponential distribution because of the long tail of pareto. Consequently, update interval has to be small to capture the fluctuations due to such short flows. That is why the performance of



(a) link capacity = 20



(b) link capacity = 40

Figure 17: Performance under traffic with variable bandwidth requests

wsp(30) degrades while *wsp*(15) is not affected. The relative performance of these schemes in the third scenario is similar to the first scenario with short and long flows. An important thing to note is that in all the scenarios the performance of *psr* is insensitive to the holding times of flows.

The behavior of *psr* is not surprising since Erlang formula is known to be applicable even when the flow holding times are not exponentially distributed and blocking probability depends only on the load, i.e., the ratio of arrival rate and service rate. For the above case of two types of flows, the aggregate arrival rate, λ, is given by $\lambda = \lambda_1 + \lambda_2$ and the mean holding time, $1/\mu$, is given by $\frac{1}{\mu} = \frac{1}{\mu_1}\frac{\lambda_1}{\lambda_1+\lambda_2} + \frac{1}{\mu_2}\frac{\lambda_2}{\lambda_1+\lambda_2}$. This heterogeneous traffic can then be treated as equivalent to homogeneous traffic with arrival rate λ, mean holding time $1/\mu$ and the corresponding load $\lambda/\mu = \lambda_1/\mu_1 + \lambda_2/\mu_2$. So for a given load, the blocking probability would be the same irrespective of the mean holding times of individual flows. That is why the performance of the theoretical scheme, *vcr* depends only on the overall offered load and not on the types of traffic. The practical scheme, *psr* also behaves similarly and hence *psr* can be employed *as is* to route flows with mixed holding times.

Varying bandwidth requests

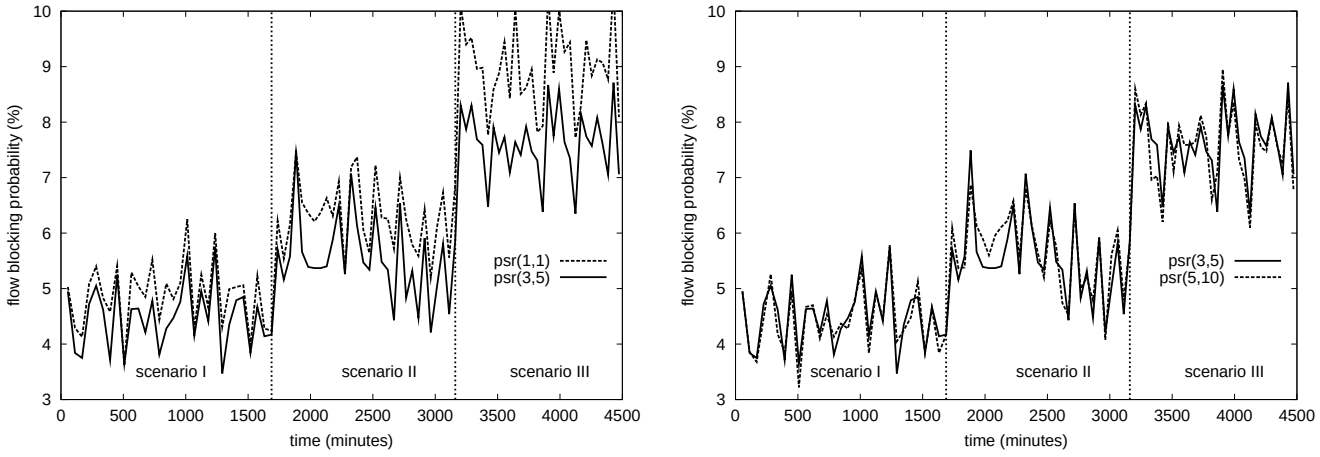
Now, consider the case of traffic with 2 types of flows, each requesting for *different amount of bandwidth* but having the same mean holding time. The bandwidth requests of flows are derived uniformly from a range: 0.5 to 1.5 for *small* flows and 1.5 to 2.5 for *large* flows, i.e., the mean bandwidth of small flows is 1 while it is 2 for large flows. The holding times of all the flows are drawn from an exponential distribution with mean 60 sec. The performance is measured varying the mix of small and large flows. Figure 17(a) shows the bandwidth blocking ratio as a function of the fraction of small flows. First thing to note is that *psr* performs poorly when the majority of flows are large. However, as the number of small flows increases, it approaches the performance of *wsp*(30). The reason is that

routing under *psr* is independent of the amount of bandwidth requested while *wsp* is conscious of the bandwidth requested. However, when the link capacity is much larger than a flow's bandwidth request, *psr* performs fine even though it is unconscious of the requested amount. To illustrate this, we increased the capacity of all links to 40 and measured the performance of both the schemes under similar load conditions as the previous case. Figure 17(b) shows that *psr* performs as well as *wsp*(30) when all the flows are large and approaches *wsp*(15) as the number of small flows increases. In the following, we argue further that when bandwidth requests are significantly smaller than the link capacity, it is not necessary for *psr* to differentiate between different bandwidth requests.

In [31], it was shown that when the capacity of a link is large, the blocking probability of a flow of type i can be approximated as follows. Suppose that type i flow requests for d_i units of bandwidth and the load of type i flows on link l is ν_l^i. The blocking probability for type i flows on link l is given by $b_l^i = \frac{d_i}{\delta} E(\frac{\sum \nu_l^i d_i}{\delta}, \frac{c_l}{\delta})$, where δ is an "equivalent rate" given by $\delta = \frac{\sum \nu_l^i d_i^2}{\sum \nu_l^i d_i}$. In other words, the ratio of blocking probabilities of flow types i and j would be same as the ratio of their bandwidth requests, i.e., $\frac{b_i}{b_j} \approx \frac{d_i}{d_j}$. This implies that $\frac{\lambda_1 b_1}{\lambda_2 b_2} = \frac{\phi_1}{\phi_2}$, i.e., the blocking rate of flows of a type is proportional to their fraction in the total offered load. Consequently, performance of a equalization based proportional routing scheme would be same with or without categorizing the flows into different classes. However, *psr* has to be extended to route flows with relatively large bandwidth requests, since it is possible that a path that is good for one bandwidth request may not be even feasible for another bandwidth request. In such a case, since the amount of bandwidth requested by a flow is known at the time of path selection, it makes sense to utilize this knowledge in categorizing them into bandwidth classes and routing them accordingly. Considering that in practice link capacities are much larger than an individual flow's bandwidth request, *psr* can be used *as is* to route heterogeneous traffic in most cases.

4.4 Sensitivity of *psr*

We now study the sensitivity of *psr* to the settings of its configurable parameters, η and $\hat{\gamma}$. These parameters control the observation period between successive computations of proportions. While η specifies the number of cycles in an observation period, $\hat{\gamma}$ gives the number of blocks permitted per path in a cycle and thus indirectly controls the length of a cycle. We have experimented with several settings of (η,$\hat{\gamma}$) and here we present the results of three different settings: $(1, 1)$, $(3, 5)$, and $(5, 10)$ in Figure 18. Two separate graphs are shown for readability. The traffic patterns and loads are varied to see the adaptivity of *psr* under different settings. In scenario I, a load of 0.35 is offered between border nodes and in scenario II, an additional load of 0.05 is offered between hot pairs only and this hot load is increased to 0.10 in scenario III. Under all settings, *psr* adapts quickly to traffic scenario changes. But *psr*$(3, 5)$ blocks lesser flows than *psr*$(1, 1)$ while no discernible difference between *psr*$(3, 5)$ and *psr*$(5, 10)$. The performance difference between *psr*$(1, 1)$ and *psr*$(3, 5)$ is more evident in scenario III where the overall offered load is high. In general, fewer the blocks permitted in a cycle, lesser the effect of proportional routing. Relatively longer cycles are needed to get a good estimate of right proportions. Also, from the perspective of stability it is better to change proportions gradually to reduce oscillations. From these results, we observe that 3 cycles and 5 blocks per



(a) (3,5) *vs* (1,1)

(b) (3,5) *vs* (5,10)

Figure 18: Performance of *psr* under different (η,$\hat{\gamma}$) settings

path per cycle seem to work fine and beyond that *psr* is relatively insensitive to its parameter settings.

4.5 Routing Stability

An essential feature of a good routing scheme is its ability to avoid routing oscillations and thus ensure stability. It was shown [35] that out-of-date information due to larger update intervals can cause route flapping in schemes such as *wsp*. When the utilization on a link is low, an update causes all the source nodes to prefer routes along this path, resulting in a rapid increase in its utilization. Similarly when the utilization is high, an update causes all the sources to shun this link and consequently its utilization decreases as the existing flows depart. This synchronization problem is inherent in any global information exchange based QoS routing schemes such as *wsp*. On the other hand, the *psr* scheme doesn't exhibit such route flapping behavior. There are two fundamental reasons for the stability of *psr*. First, in *psr* each source performs routing based on its own *local view* of the network state. Routing based on such a "customized view" avoids the undesirable *synchronized mass reaction* that is inherent in QoS routing scheme based on a global view. Second, *psr* does proportional routing with a proportion assigned to a path reflecting its quality. A relatively better path is favored by sending larger proportion of traffic to it. It doesn't pick just one "best" path. The psr can also cause higher fluctuation occasionally at the end of a cycle due to making some paths ineligible and routing all the load along one or a few eligible paths. However, as proportions stabilize, duration of such fluctuations tend be smaller. Considering all this we claim that a localized proportional routing scheme such as *psr* is intrinsically more stable than a global best-path routing scheme such as *wsp*.

4.6 Routing Overhead

We now take a close look at the amount of overhead involved in these two routing schemes. This overhead can be categorized into path selection overhead and information collection overhead. We discuss these two separately in the following.

The *wsp* scheme selects a path by first pruning the links with insufficient available bandwidth and then performing a variant of Dijkstra's algorithm on the resulting graph to find the shortest path with maximum bottleneck bandwidth. This takes at least $O(E \log N)$ time where N is the number of nodes and E is the total number of links in the network. Assuming precomputation of a set of paths R to each destination to avoid searching the whole graph for path selection, it still need to traverse all the links of these precomputed paths. This amounts to an overhead of $O(L)$, where L is the total number of links in the set R. On the other hand, the path selection in *psr* is simply an invocation of $wrrps$ whose worst case complexity is $O(|R|)$ which is much less than $O(L)$ for *wsp*.

Now consider the information collection overhead. In *wsp*, each source acquires a network-wide view on the status of links through link state updates. Every router is responsible for maintaining QoS state and generating updates about all the links adjacent to it. These updates are sent either periodically or after a significant change in the resource availability since the last update. They are propagated to all the routers in the network through flooding. As in OSPF [22] each router is responsible for maintaining a consistent QoS state database. This incurs both communication and processing overhead. In contrast, the routers employing *psr* scheme do not exchange any such updates and thus completely do away with this overhead. Only source routers need to keep track of route level statistics and recompute proportions after every observation period. Statistics collection in *psr* involves only increment and decrement operations costing only constant time per flow. The proportion computation procedure in *psr* itself is extremely simple and costs no more than $O(|R|)$.

5 Related Work

The problem of QoS routing has been addressed in several contexts, a survey of which can be found in [15]. The work more relevant to ours is the distributed routing scheme proposed in [16] where a set of multiple paths are probed in parallel, using tickets, for a satisfactory path. However, this approach requires the distribution and processing of these tickets by intermediate nodes. Minimum interference routing [12] is a scheme proposed recently that selects a path that interferes least with the routing of future flows. While this scheme provides good routing performance, it has significant computational overhead. The proportional routing approach presented in this paper achieves the similar effect by gradually adapting the flow proportions assigned to paths based on their blocking probabilities which is an indirect measure of interference of paths.

It is interesting to contrast *psr* with some of the dynamic routing schemes proposed in the context of telephone networks. Here we consider two such schemes based on sticky routing and learning automata that make use of the feedback information regarding flow admission or rejection for routing future flows.

Table 2: Comparison of blocking under various routing schemes

Topology	Scenario	*psr*	*rsr*	$L_{R-\epsilon P}$	L_{R-P}
fork (3)	$\nu = 45$	5.12	7.67	8.06	6.51
fork (3)	$c_2 = 10, c_3 = 5, \nu_s = 25$	6.38	9.72	9.18	8.35
kite	$\nu_{s_1} = 22, \nu_{s_2} = 22$	4.24	5.59	5.84	5.12
kite	$\nu_{s_1} = 25, \nu_{s_2} = 15$	2.12	3.58	3.81	3.65
duck	$\nu = 30$	4.57	9.13	7.35	6.88
fish	$\nu_{s_1} = 20, \nu_{s_2} = 15, \nu_{s_3} = 5, \nu_{s_4} = 10$	1.02	4.40	4.26	3.07

5.1 Sticky Random Routing

The *dynamic alternative routing* (dar) is a well known routing scheme [9] where a source always tries the direct one-link path to the destination first and in case of a crankback chooses a two-link path using *sticky random routing* (srr). Since in our setting we do not consider re-routing, the *srr* scheme (equivalent to *dar* with a dummy direct link) is used for comparison. The *srr* scheme remembers a path known as *preferred* path for each destination. A flow to a destination is always routed through its corresponding preferred path. If the connection setup is successful, the preferred path remains same. But in case of a failure, the flow is blocked and a new preferred path is chosen randomly from set of feasible paths to that destination excluding the current preferred path. The *srr* scheme essentially sticks to a path as long as it can accommodate offered traffic.

The analysis of *dar* presented in [9] observes that *dar* equalizes the blocking rates over two-link paths for each source destination pair. It claims that overflow streams, i.e., flows directed to two-link paths, under *dar* can be modeled as if they arise from proportional routing, with proportions depending on the blocking rates of links. But it also cautions that the approximation procedure used in the analysis could break down if the overflow is large and needs to be spread over a number of alternatives. This is precisely the case with networks like Internet that may have more than one minhop path and many alternative paths between each source-destination pair.

5.2 Learning Automata based Routing

An application of automata to the routing problem is given by Narendra and Mars [24]. The incoming flows are offered to a path r according to a probability distribution p_r, which is updated using feedback information regarding flow admission or rejection. These schemes reward a path on which a flow is successful and punish a path on which a flow fails. If a route i is chosen at time n and the flow is successful, then updating is

$$p_i(n+1) = p_i(n) + a(1 - p_i(n))$$

$$p_j(n+1) = (1-a)p_j(n) \quad j \neq i$$

while if the flow fails

$$p_i(n+1) = (1-\epsilon)p_i(n))$$

$$p_j(n+1) = \frac{\epsilon}{r-1} + (1-\epsilon)p_j(n) \quad j \neq i$$

where a and ϵ are adjustable parameters, $0 < a < 1$, $0 < \epsilon < 1$ with ϵ small compared with a, and a is itself usually small, so that the updating is gradual. Under certain assumptions [23, 36] show that $L_{R-\epsilon P}$ automata tends to approximately equalize blocking probabilities, b_r, while L_{R-P} automata for which $\epsilon = a$ in the above equalizes blocking rates ($p_r b_r$). One problem with these schemes is that no account is taken of the length of the path.

5.3 Comparison with *psr*

The above schemes are compared with the *psr* scheme by simulating them and observing their performance under different settings. The parameters ϵ and a in $L_{R-\epsilon P}$, were set to 0.01 and 0.02 respectively. Similarly for $L_{R-\epsilon P}$ the settings were $\epsilon = a = 0.01$. Table 2 compares the overall blocking under these schemes for different topologies and load conditions. The capacities of all bottleneck links are set to 20 except in one case where $c_2 = 10$ and $c_3 = 5$. It can be seen that in all cases *psr* performs better than the other schemes.

6 Conclusions and Future Work

This paper focused on *localized* QoS routing schemes where the edge routers make routing decisions using only "local" information. Such an approach to proportional routing has several advantages: minimal communication overhead, no processing overhead at core routers, and easy deployability. As a first step towards designing a simple localized scheme, we developed *virtual capacity based routing* (vcr), a theoretical scheme based on the notion of *virtual capacity* of a route. We then proposed *proportional sticky routing* (psr), an easily realizable approximation of *vcr* and analyzed its performance. We demonstrated through extensive simulations that *psr* scheme is indeed simple, stable, and adaptive. We have also shown that the proposed scheme is insensitive to the durations of flows and also that when the link capacities are significantly larger than bandwidth requests of flows, *psr* scheme can be employed *as is* to route heterogeneous flows. We have compared the performance of *psr* with *wsp* and shown that *psr* performs as well as *wsp* even at smaller update intervals. In particular, we found that *psr* performs better than *wsp* when higher load is offered from fewer sources and when the flows are of shorter duration and smaller bandwidth. We conclude that the *psr* scheme, with low overhead and comparable performance, is a viable alternative to global QoS routing schemes such as *wsp*.

The localized approach to proportional routing is simple and has several important advantages. However it has a limitation that routing is done based solely on the information collected locally. A network node under localized QoS routing approach can judge the quality of paths only by routing some traffic along them. It would have no knowledge about the state of the rest of the network. While the proportions for paths are adjusted to reflect the changing qualities of paths, the candidate path set itself remains static. To ensure that the localized scheme adapts to varying network conditions, many feasible paths have to be made candidates. It is not possible to preselect a few good candidate paths statically. Hence it is desirable to supplement localized proportional routing with a mechanism

k	:	total number of paths in set R^{elg}.
r_i	:	path associated with index i.
w_{r_i}	:	weight associated with path r_i.
n_{r_i}	:	number of times path r_i was selected.
l	:	run length of the most recently selected path.
$\mathcal{W}_i$	:	$w_{r_i} + w_{r_{i+1}} + \cdots + w_{r_k}$.
$\mathcal{N}_i$	:	$n_{r_i} + n_{r_{i+1}} + \cdots + n_{r_k}$.

(a) notation

```
1.   PROCEDURE wrrps()
2.     For i = 1, 2, ..., k
3.       If l*W_{i+1} < w_{r_i} and W_{i+1}*n_{r_i} <= w_{r_i}*N_{i+1}
4.         break
5.       Set W_{i+1} = W_{i+1} + w_{r_i} - w_{r_{i+1}}
6.       Set N_{i+1} = N_{i+1} + n_{r_i} - n_{r_{i+1}}
7.       Swap r_i and r_{i+1}
8.       Set l = 0
9.     Set n_{r_0} = n_{r_0} + 1; N_{r_0} = N_{r_0} + 1;
10.    Set l = l + 1
11.    Return r_0
12.  END PROCEDURE
```

(b) path selection

Figure 19: The *wrrps* procedure

that dynamically selects a few good candidate paths. We proposed such a hybrid approach in [28] where a few widest disjoint paths are selected as candidates based on infrequently globally exchanged link state metrics and flows are proportioned among these candidate paths based on locally collected path state metrics. We have also extended our proportional routing approach to provide hierarchical routing across multiple areas in a large network. More details can be found in [29].

Appendix: Weighted Round Robin Procedure for Path Selection

Given a set R^{elg} of eligible paths and their associated proportions $\{\alpha_r, r \in R^{elg}\}$, *wrrps* picks a path $r \in R^{elg}$ based on its weight, $w_r = \frac{\alpha_r}{\sum_{s \in R^{elg}} \alpha_s}$. Instead of using a probabilistic method such as picking a path r with probability w_r, we opt to employ a deterministic algorithm to ensure that flow proportions are preserved within as small a time window as possible. This is implemented by using a deterministic sequence of paths which has the property that the paths are distributed periodically with a frequency which closely approximates the prescribed flow proportions. This is implemented by generating a sequence of paths that preserves flow proportions within as small a window as possible. This sequence is generated by *wrrps* on the fly: for an incoming flow, *wrrps* generates the next path in the sequence and routes the flow along the path.

The *wrrps* procedure is shown in Figure 19. It keeps track of the number of times each path was selected (n_{r_i}) and the run length (l) of the most recently selected path. It maintains an ordered list of paths and the first path in the list is selected as long it satisfies both the following constraints: 1) its weight is more than its run length times the weight of the rest of paths ($l\mathcal{W}_1 < w_{r_0}$); 2) ratio of number of times it was selected and the number of times all others were selected is less than or equal to the ratio of its weight and weight of the rest of paths ($\mathcal{W}_1 n_{r_0} \leq w_{r_0}\mathcal{N}_1$). Otherwise this path is pushed down the order and the run length is reset to 0. Then it returns the first path in the list. A sample *wrrps* generated sequence where the current eligible set R^{elg} has four paths r_1, r_2, r_3 and r_4 with

weights $1/2$,$1/4$,$1/8$, and $1/8$ respectively is: $r_1\ r_2\ r_1\ r_3\ r_1\ r_2\ r_1\ r_4\ r_1\ r_2\ r_1\ \mathbf{r_3}$. This sequence has the property that in every window of size 2 there is an r_1 and an r_2 in every window of size 4. Similarly one r_3 and one r_4 in all windows of size 8. Assuming that up to the last r_1 are the paths chosen so far, the next path selected on the fly by the *wrr* path selector would be r_3. Note also that every time the eligible path set R^{elg} changes, a new sequence is generated, and flows arriving thereafter are thus routed according to this new sequence.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, S. Tripathi, "Quality of Service Based Routing: A Performance Perspective", ACM SIGCOMM 1998.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, S. Tripathi, "Improving QoS Routing Performance under Inaccurate Link State Information", ITC'16, June 1999.

[3] G. Ash, *Dynamic Routing in Telecommunications Networks*, McGraw-Hill, 1998.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering over MPLS," RFC-2702, September 1999.

[5] R.Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, "A Framework for Multiprotocol Label Switching," IETF Internet Draft, Work in Progress, September 1999.

[6] J. Chen, P. Druschel, and D. Subramanian, "A New Approach to Routing with Dynamic Metrics", IEEE INFOCOM 1999.

[7] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick, "A Framework for QoS-Based Routing in the Internet", *Work in Progress*, Internet Draft, July 1998.

[8] R. F. Farmer and I. Kaufman, "On the Numerical Evaluation of Some Basic Traffic Formulae," in *Networks, Vol 8.*, pp. 153–186, John Wiley and Sons, Inc., 1978.

[9] R.J. Gibbens, F.P. Kelly, and P.B. Key, "Dynamic Alternative Routing: Modelling and Behaviour", Teletraffic Science, pp. 1019-1025, Elsevier, Amsterdam, 1989.

[10] R. Guerin, S. Kamat, A. Orda, T. Przygienda, D. Williams, "QoS Routing Mechanisms and OSPF Extensions", *Work in Progress*, Internet Draft, March 1997.

[11] R. Guerin, A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms", IEEE INFOCOM 1997.

[12] Murali Kodialam, and T. V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffic Engineering", INFOCOM 2000.

[13] F.P. Kelly, "Routing in Circuit-Switched Networks: Optimization, Shadow Prices and Decentralization", Advances in Applied Probability 20, 112-144, 1988.

[14] F.P. Kelly, "Routing and capacity Allocation in Networks with Trunk Reservation", Mathematics of Operations Research, 15:771-793, 1990.

[15] Shigang Chen, Klara Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions", IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video, Vol. 12, No. 6, November-December 1998, pp. 64-79.

[16] Shigang Chen, and Klara Nahrstedt, "Distributed QoS Routing with Imprecise State Information", ICCCN'98, October 1998.

[17] J.S. Kaufman, "Blocking in a Shared Resource Environment," IEEE Trans. Commun. vol. COM-29, pp. 1474-1481, 1981.

[18] F.P. Kelly, "Dynamic Routing in Stochastic Networks", In *Stochastic Networks*, ed. F.P. Kellyand R.J. Williams, Springer-Verlag, 169-186, 1995.

[19] P.B. Key, and G.A. Cope, "Distributed Dynamic Routing Schemes", IEEE Communications Magazine, pp. 54-64, Oct 1990.

[20] Q. Ma, P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees", IEEE ICNP 1997.

[21] D. Mitra, and J.B. Seery, "Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks", IEEE Trans. on Communications, vol. 39, no. 1, pp. 102-116, January 1991

[22] J. Moy, "OSPF Version 2", Request For Comments 2178, Internet Engineering Task Force, July 1997.

[23] K.S. Narendra and M.A.L. Thathachar, "On the Behavior of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing", IEEE Trans. on Systems, Man and Cybernetics, vol. SMC-10, no.5, pp. 262-269, May 1980.

[24] K.S. Narendra and P. Mars, "The Use of Learning Algorithms in Telephone Traffic Routing - A Methodology", Automatica, vol. 19, no. 5, pp. 495-502, 1983.

[25] S. Nelakuditi, R.P. Tsang, Z-L. Zhang, "Quality-of-Service Routing without Global Information Exchange", IWQOS 1999.

[26] S. Nelakuditi, Z-L. Zhang, R.P. Tsang, "Adaptive Proportional Routing: A Localized QoS Routing Approach", IEEE INFOCOM'00, March 2000.

[27] S. Nelakuditi, S. Varadarajan, and Z-L. Zhang, "On Localized Control in Quality-of-Service Routing", To Appear in IEEE Transactions on Automatic Control, Special Issue on Systems and Control Methods for Communication Networks.

[28] S. Nelakuditi, and Z.-L. Zhang, "On Selection of Paths for Multipath Routing", IWQOS'01, June 2001.

[29] S. Nelakuditi, "Localized Approach to Providing Quality-of-Service," Ph.D Dissertation, Department of Computer Science, University of Minnesota, October 2001.

[30] J.W. Roberts, "Teletraffic Models for the Telecom 1 Integrated Services Network," in Proc. Internet Teletraffic Congress-10, Session 1.1, paper #2.

[31] J. Roberts, U. Mocci, and J. Virtamo, "Broadband Network Teletraffic," LNCS 1155, Springer Verlag, 1996.

[32] E. Rosen, A. Viswanathan, and R. Callon, "Multi-Protocol Label Switching Architecture", *work in progress*, Internet Draft draft-ietf-mpls-arch-06.txt, August 1999.

[33] K.W. Ross, "Multiservice Loss Models for Broadband Telecommunication Networks", Springer-Verlag, 1995.

[34] A. Shaikh, J. Rexford, K. Shin, "Evaluating the Overheads of Source-Directed Quality-of-Service Routing", ICNP 1998.

[35] A. Shaikh, J. Rexford, K. Shin, "Load-Sensitive Routing of Long-Lived IP Flows", ACM SIGCOMM 1998.

[36] P.R. Srikantakumar and K.S. Narendra, "A Learning Model for Routing in Telephone Networks", SIAM J. Control and Optimization, vol. 20, no. 1, pp. 34-57, January 1982.

[37] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-Service Routing using Maximally Disjoint Paths", IWQOS 1999.

[38] Z. Wang, J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications", IEEE JSAC Sept 1996

[39] E. W. Zegura, K.L. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," IEEE INFOCOM 1996.

[40] Fang Hao and Ellen Zegura, "On Scalable QoS Routing: Performance Evaluation of Topology Aggregation", IEEE INFOCOM 2000.

[41] Z. Zhang, C. Sanchez, B. Salkewicz, E. Crawley, "Quality of Service Extensions to OSPF", *Work in Progress*, Internet Draft, September 1997.